1/5/1
DIALOG(R)File 351:Derwent WPI

013196222 **Image available**
WPI Acc No: 2000-368095/200032
XRPX Acc No: N00-275535

Information processing system for high security IC card has CPU that executes given process in accordance with program to process data, which includes one or more data process devices

Patent Assignee: HITACHI LTD (HITA)
Inventor: FUKUZAWA Y; KAMINAGA M; OKI M; OKUHARA S; OHKI M
Number of Countries: 030 Number of Patents: 006
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 1006492	A1	20000607	EP 99123518	A	19991125	200032	B
JP 2000165375	A	20000616	JP 98338779	A	19981130	200036	
CN 1255692	A	20000607	CN 99125831	A	19991130	200046	
KR 2000052392	A	20000825	KR 9953692	A	19991130	200121	
US 6408075	B1	20020618	US 99449537	A	19991129	200244	
			US 2000525014	A	20000314		
TW 466393	A	20011201	TW 99120917	A	19991201	200252	

Priority Applications (No Type Date): JP 98338779 A 19981130
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 1006492	A1	E	42	G07F-007/10	
Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI					
JP 2000165375	A		28	H04L-009/10	
CN 1255692	A			G06K-019/07	
KR 2000052392	A			G06K-019/07	
US 6408075	B1			H04L-009/28	Div ex application US 99449537
TW 466393	A			G06C-001/00	

Abstract (Basic): EP 1006492 A1

NOVELTY - A CPU (201) executes a given process in accordance with a program to process a data. The program includes one or more data process devices, each of which has a process instruction for giving an execution instruction to the CPU. The data is a combination of a normal data and its bit inverted data, which include data under a normal process and data obtained by inverting bits of the data under the normal process.

USE - In IC card - smart card.

ADVANTAGE - Provides tamper resistant device of high security. Reduced correlation or dependency between data processing and its consumption current of an IC chip.

DESCRIPTION OF DRAWING(S) - The drawing shows a hardware structure of an IC card chip.

CPU (201)

pp; 42 DwgNo 1,2/31

Title Terms: INFORMATION; PROCESS; SYSTEM; HIGH; SECURE; IC; CARD; CPU; EXECUTE; PROCESS; ACCORD; PROGRAM; PROCESS; DATA; ONE; MORE; DATA; PROCESS; DEVICE
Derwent Class: T01; T04; T05; U21
International Patent Class (Main): G06C-001/00; G06K-019/07; G07F-007/10; H04L-009/10; H04L-009/28
International Patent Class (Additional): G06F-001/00; G06F-007/552; G06K-019/073; G09C-001/00
File Segment: EPI

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－165375
（P2000－165375A）
(43)公開日　平成12年６月16日(2000.6.16)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'（参考）
Ｈ０４Ｌ　9/10		Ｈ０４Ｌ　9/00	６２１Ｚ	５Ｂ０３５
Ｇ０６Ｋ　19/073		Ｇ０９Ｃ　1/00	６６０Ａ	５Ｊ１０４
Ｇ０９Ｃ　1/00	６６０	Ｇ０６Ｋ　19/00	Ｐ	

審査請求　未請求　請求項の数34　ＯＬ　（全 28 頁）

(21)出願番号　特願平10－338779

(22)出願日　平成10年11月30日(1998.11.30)

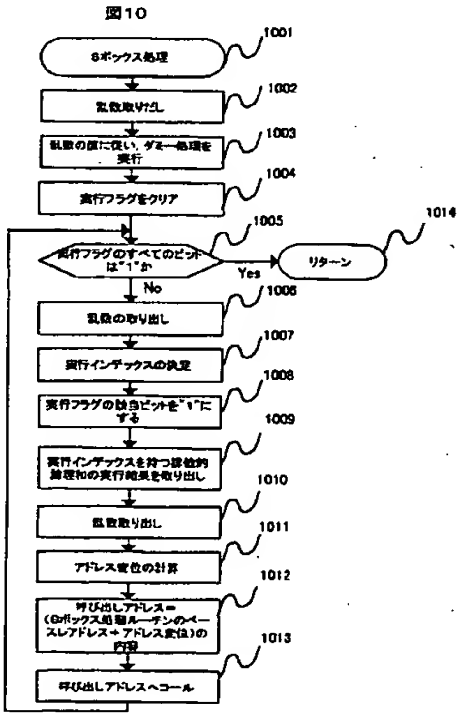
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　大木　優
東京都国分寺市東恋ケ窪一丁目280番地
株式会社日立製作所中央研究所内
(72)発明者　福澤　寧子
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
(74)代理人　100068504
弁理士　小川　勝男

最終頁に続く

(54)【発明の名称】　情報処理装置、ＩＣカード

(57)【要約】
【課題】　ICカード用チップでの処理のデータとICカード用チップの消費電流との関連性を減らす。
【解決手段】　ICカード用チップの処理順序をランダムに入れ替えたり、無駄なダミー処理を追加することにより、処理データとICカード用チップの消費電流との関連性を減らす。ICカード用チップでの処理のデータとICカード用チップの消費電流との関連性を減らすことを可能とする。

図10



1

【特許請求の範囲】

【請求項１】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い所定の処理を実行しデータ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、正常処理のデータとそのビットの値を反転したデータとからなる正常データ及びビット反転データの組みをデータとして有することを特徴とする情報処理装置。

【請求項２】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、同一の命令で、正常処理のデータとそのビットの値を反転したデータとを処理する正常データ及びビット反転データ処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項３】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、正常処理命令と同じ命令コマンドを有し、かつ正常処理命令において処理されるデータのビットの値を反転したデータを処理する命令を有する正常データ命令及びビット反転データ命令処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項４】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、正常処理の結果のビットの値を反転したデータを結果として生成するビット反転データ生成手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項５】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段として、異なる処理データを複数繰り返して処理する繰り返しデータ処理手段を含む、情報処理装置において、繰り返しデータ処理手段をランダムに実行する繰り返しランダム実行処理手段をデータ処理手段として有することを特徴とする情報処理装置。

2

【請求項６】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、プログラム内にプログラムの該当処理に影響を与えないダミー実行処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項７】請求項５または６に記載の情報処理装置において、ダミー実行処理手段と繰り返しデータ処理手段とをランダムに実行するダミー実行及び繰り返しランダム実行処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項８】請求項１から７に記載の情報処理装置において、ビット単位のデータを入れ替える転置処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項９】請求項１から７に記載の情報処理装置において、バイト単位のデータを入れ替える換字処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項１０】請求項１から７に記載の情報処理装置において、入力データと暗号鍵データの排他的論理和を行う排他的論理和処理手段と入力データのビットを反転したデータと暗号鍵の排他的論理和を行うビット反転データ排他的論理和処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項１１】請求項１から７に記載の情報処理装置において、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段と、入力データのビットを反転したデータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段とをデータ処理手段として有することを特徴とする情報処理装置。

【請求項１２】請求項１から７に記載の情報処理装置において、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段と、入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項１３】請求項１から７に記載の情報処理装置において、入力データと暗号鍵データの排他的論理和を行う排他的論理和処理手段と、入力データを反転したデータと暗号鍵の排他的論理和を行うビット反転データ排他

的論理和処理手段、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段、入力データのビットを反転したデータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段、入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段、をデータ処理手段として有することを特徴とする情報処理装置。

【請求項１４】プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、入力データと暗号鍵を使って、べき乗乗算を繰り返してべき乗剰余計算を行う手段を有するデータ処理手段を有する情報処理装置において、暗号鍵のビットの値に関わらず、入力データとこれまで計算した結果を乗算剰余する乗算剰余処理手段と、暗号鍵のビットが１ならば、乗算剰余処理手段の結果を使い、０であれば、乗算剰余処理手段の結果を無視する乗算剰余処理結果選択処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項１５】プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、複数ビットの値に対応して入力データのべき乗剰余計算した結果であるビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、暗号鍵の複数ビットの値毎に、ビット値対応入力データべき乗剰余計算処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段、をもつ入力データと暗号鍵を使ってべき乗剰余計算を行うデータ処理手段を有する情報処理装置において、ビット対応乗算剰余計算処理手段を繰り返し処理しているあるタイミングにおいて、ビット値対応入力データべき乗剰余値を変更するビット値対応入力データべき乗剰余値変更処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項１６】請求項１５に記載の情報処理装置において、ビット値対応入力データべき乗剰余値変更処理手段でビット値対応入力データべき乗剰余値に変更を加えた処理結果を変更しない値に戻すビット値対応入力データべき乗剰余値変更処理逆変換処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項１７】請求項１５に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項１８】請求項１７に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると１になる二つの値vとuの中の一方であるv、あるいはvのべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段とビット値対応入力データべき乗剰余値変更処理手段において、vを剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果にuを剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段をビット値対応入力データべき乗剰余値変更部とビット値対応入力データべき乗剰余値変更処理逆変換処理手段として有することを特徴とする情報処理装置。

【請求項１９】請求項１８に記載の情報処理装置において、上記ビット値対応入力データべき乗剰余値変更処理手段は、剰余演算の法Ｎにおいて掛けると１になる二つの値として、２とN+1/2を使うことを特徴とする情報処理装置。

【請求項２０】請求項１８に記載の情報処理装置において、暗号鍵をＭビットの組として表し、そのビットが０あるいは１であるすべての組み合わせに対して、その値を、入力データにべき乗乗算してビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、あるタイミングで、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると１になる二つの値vとuの中の一方であるv、あるいはvのべき乗を該当剰余演算の法Ｎにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、暗号鍵の複数ビットＭの値毎に、ビット値対応入力データべき乗剰余値変更処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段、ｖを剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果にuを剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項２１】請求項２０に記載の情報処理装置において、あるタイミングで、ビット値対応入力データべき乗剰余値に、ｖとして２のランダムなべき乗を該当剰余演

算の法Ｎにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、２を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に(n+1)/2を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項２２】請求項１７に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、２の任意数のべき乗、あるいは２の８乗の任意数のべき乗を剰余演算の法に掛けた数を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項２３】プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、剰余演算を行うデータ処理手段を有する情報処理装置において、剰余演算の法を変更する剰余演算法変更処理手段と、剰余演算の右辺の値を変更する剰余演算右辺変更処理手段、剰余演算法変更処理手段で変更した剰余演算の法を変更前の計算結果に戻す剰余演算法変更処理逆変換処理手段、をデータ処理手段として有することを特徴とする情報処理装置。

【請求項２４】請求項２３の剰余演算法変更処理手段と剰余演算右辺変更処理手段、剰余演算法変更処理逆変換処理手段において、法の変更手段として、整数倍を掛ける剰余演算法変更処理手段と、剰余演算の結果を剰余演算法変更処理手段で使用した整数を剰余演算の右辺の値に掛ける剰余演算右辺変更処理手段、計算結果に戻す手段として、剰余演算の結果を剰余演算法変更処理手段で使用した整数でわる剰余演算法変更処理逆変換処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項２５】請求項２４の剰余演算法変更処理手段と剰余演算右辺変更処理手段、剰余演算法変更処理逆変換処理手段において、整数倍として２の任意数のべき乗、あるいは、２の８乗の任意数のべき乗を使用する剰余演算法変更処理手段と、剰余演算の右辺の値を、剰余演算法変更処理手段で使用した整数倍として２の任意数のべき乗、あるいは、２の８乗の任意数のべき乗で掛ける剰余演算右辺変更処理手段、計算結果に戻す手段として、剰余演算の結果を剰余演算法変更処理手段で使用した整数倍として２の任意数のべき乗、あるいは、２の８乗の任意数のべき乗でわる剰余演算法変更処理逆変換処理手段、をデータ処理手段として有することを特徴とする情報処理装置。

【請求項２６】プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、剰余演算を行うデータ処理手段を有する情報処理装置において、ビット値対応入力データべき乗剰余値を変更するビット値対応入力データべき乗剰余値変更処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項２７】請求項２６に記載の情報処理装置において、ビット値対応入力データべき乗剰余値変更処理手段でビット値対応入力データべき乗剰余値に変更を加えた処理結果を変更しない値に戻すビット値対応入力データべき乗剰余値変更処理逆変換処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項２８】請求項２６に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項２９】請求項２８に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると１になる二つの値vとuの中の一方であるv、あるいはvのべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段とビット値対応入力データべき乗剰余値変更処理手段ｖを剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果にuを剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段をビット値対応入力データべき乗剰余値変更部とビット値対応入力データべき乗剰余値変更処理逆変換処理手段として有することを特徴とする情報処理装置。

【請求項３０】請求項２９に記載の情報処理装置において、上記ビット値対応入力データべき乗剰余値変更処理手段において剰余演算の法Ｎにおいて掛けると１になる二つの値として、２とN+1/2を使うことを特徴とする情報処理装置。

【請求項３１】請求項２９に記載の情報処理装置において、暗号鍵をＭビットの組として表し、そのビットが０あるいは１であるすべての組み合わせに対して、その値を、入力データにべき乗乗算してビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、あるタイミングで、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると１になる二つの値vとuの中の一方であるv、あるいはvのべき乗を該当剰余演算の法Ｎにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手

段、暗号鍵の複数ビットMの値毎に、ビット値対応入力データべき乗剰余値変更処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段ｖを剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果にuを剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項３２】請求項３１に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、vとして２のランダムなべき乗を該当剰余演算の法Ｎにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、２を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に(n+1)/2を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項３３】請求項２６に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、２の任意数のべき乗、あるいは２の８乗の任意数のべき乗を剰余演算の法に掛けた数を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項３４】上記情報処理装置は、ＩＣカードであることを特徴とする請求項１から３３に記載の情報処理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、情報処理装置、特に、機密性の高いICカードなどの耐タンパ装置に関するものである。

【０００２】

【従来の技術】ICカードは、主に、勝手に書き換えられない情報の保持や秘密情報である暗号鍵を使ったデータの暗号化や暗号文の復号化を行うために使われる装置である。ICカードは、電源を持っていないため、リーダライタに差し込まれると、電源の供給を受け、動作可能となる。動作可能になると、リーダライタからコマンドを受けて、コマンドに従い、データの転送を行う。ICカードの一般的な解説は、オーム社出版電子情報通信学会編水沢順一著「ICカード」などにある。

【０００３】ICカードの構成は、図１に示すように、カード１０１の上に、ICカード用チップ１０２を搭載したものである。一般にICカードは、接点を持ち、接点を通して、リーダーライタから電源の供給やリーダライタとのデータの通信を行う。

【０００４】ICカード用チップの構成は、基本的にマイクロコンピュータと同じような構成である。その構成は、図２に示すように、中央演算装置２０１、記憶装置２０４、入出力ポート２０７、コ・プロセッサ２０２からなる。中央処理装置２０１は、論理演算や算術演算などを行う装置であり、記憶装置２０４は、プログラムやデータを格納する装置である。入出力ポートは、リーダライタと通信を行う装置である。コ・プロセッサは、剰余演算を行うための特別な演算装置であり、非対称暗号であるRSAの演算などに用いられる装置である。ICカード用プロセッサの中には、コ・プロセッサを持たないものも多くある。データバス２０３は、各装置を接続するバスである。

【０００５】記憶装置２０４は、ROM(Read Only Memory)やRAM(Random Access Memory)、EEPROM(Electrical Erasable Programmable Read Only Memory)などからなる。ROMは、変更できないメモリであり、主にプログラムを格納するメモリである。RAMは自由に書き換えができるメモリであるが、電源の供給が中断されると、記憶している内容が消えてなくなる。ICカードがリーダライタから抜かれると電源の供給が中断されるため、RAMの内容は、保持できなくなる。EEPROMは、電源の供給が中断されてもその内容を保持することができるメモリである。書き換える必要があり、ICカードがリーダライタから抜かれても、保持するデータを格納するために使われる。例えば、プリペイドカードでのプリペイドの度数などは、使用するたびに書き換えられ、かつリーダライタか抜かれてもデータを保持する必要があるため、EEPROMで保持される。

【０００６】ICカードは、プログラムや重要な情報がICカード用チップの中に密閉されているため、重要な情報を格納したり、カードの中で暗号処理を行うために、使われている。ICカードでの暗号処理の解読の難しさは、暗号アルゴリズムの解読の困難さと同じと考えられていた。しかし、ICカードが暗号処理を行っている時の消費電流を観測し、解析することにより、暗号アルゴリズムの解読より容易に暗号処理の内容や暗号鍵が推定される可能性が示唆されている。消費電流は、リーダライタから供給されている電流を測定することにより観測することができる。これは、John Wiley & sons社 W.Rankl & W.Effing著 「Smart Card Handbook」の8.5.1.1 Passive protective mechanisms(２６３ページ)にこのような危険性が記載されている。

【０００７】ICカード用チップを構成しているCMOSは、出力状態が１から０あるいは０から１に変わった時に電流を消費する。特に、データバス２０３のバスは、大きな電気容量を持つため、バスの値が１から０あるいは０から１に変わると、大きな電流が流れる。そのため、消費電流を観測すれば、ICカード用チップの中で、何が動作しているか分かる可能性を示唆している。

【０００８】図５は、ICカード用チップの１サイクルでの消費電流の波形を示したものである。処理しているデータの依存して、電流波形が５０１や５０２のように異

なる。このような差は、バス２０３を流れるデータや中央演算装置２０１で処理しているデータに依存して生じる。

【０００９】コ・プロセッサ２０２は、中央演算装置と並列に、長いビット、例えば、５１２ビットの剰余演算を行うことができる。そのため、中央演算装置の消費電流とは異なった消費電流波形を長い時間、観測することができる。その特徴的な波形を観測すれば、コ・プロセッサの動作回数を容易に測定することができる。コ・プロセッサの動作回数が暗号鍵と何らかの関係があるならば、コ・プロセッサの動作回数から暗号鍵を推定できる可能性がある。

【００１０】また、コ・プロセッサでの演算内容が、暗号鍵に依存した偏りがあると、その偏りが消費電流から求められ、暗号鍵が推定される可能性がある。

【００１１】中央演算装置でも同様である。暗号鍵のビットの値は決まっているため、処理するデータを変更して、消費電流を観測することにより、暗号鍵のビットの値の影響が観測できる可能性がある。これらの消費電流の波形を統計的に処理することにより、暗号鍵を推定できる可能性がある。

【００１２】

【発明が解決しようとする課題】本発明の課題は、ICカード用チップでのデータ処理と消費電流との関連性を減らすことである。消費電流とチップの処理との関連性が減れば、観測した消費電流の波形からICカードチップ内での処理や暗号鍵の推測が困難になる。本発明の着眼点は、ICカードチップでの処理手順をランダムにすることや、ダミー処理を挿入することにより、消費電流の波形から、処理や暗号鍵の推測を困難にするものである。

【００１３】

【課題を解決するための手段】ICカード用チップに代表される耐タンパ装置は、プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置として考えることができる。本発明では、処理しているデータとICカード用チップの消費電流の関連性を減らすための方法の一つとして、処理するデータを正常処理のデータとそのビットの値を反転したデータの両方を使用する。そして、ビット反転のデータと正常データを、同一の命令で、正常処理のデータとビットの値を反転したデータとを処理することにより、データバス上の値などが“０”から“１”、あるいは“１”から“０”なる回数を一定にする。データバスでは、値が反転したときに電流を大きく消費するため、“１”と“０”の反転回数が一定になることにより、電流の消費する回数が一定になり、データ処理と消費電流の関連性を減らすことができる。

【００１４】正常データとビット反転データを同じように処理するための別の方法として、同一ルーチンで正常データとビット反転データを処理できない場合は、正常処理命令と同じ命令コマンドを有し、ビット反転したデータを処理する命令を有する処理ルーチンを使用する。さらに、常に処理した結果でも正常データとビット反転データを生成することを行うようにし、正常データとビット反転データを常に同じように処理するようにする。

【００１５】データ処理と消費電流の関連性を減らす別の方法は、処理するデータだけ異なる繰り返し処理があれば、一定の順番でデータを取り出し処理するのではなく、処理順番をランダムにする方法である。また、別の方法は、プログラムの該当処理に影響を与えないダミー実行処理を入れることにより、どこで何をしているかが一定にならないようにする方法である。ダミー実行処理手段と繰り返しデータ処理手段とをランダムに実行することを組み合わせることは、さらに、データ処理と消費電流の関連性を減らすために有効である。

【００１６】正常データとビット反転データを使う方法や、ダミー実行処理手段と繰り返しデータ処理手段を使う方法は、暗号アルゴリズムで使われるビット単位のデータを入れ替える転置処理手段やバイト単位のデータを入れ替える換字処理手段のデータ処理と消費電流の関連性を減らすために、特に有効である。

【００１７】実際、DESなどの転置や換字を使う暗号では、排他的論理和を使うことが多いので、入力データと暗号鍵データの排他的論理和を行う排他的論理和処理手段と入力データのビットを反転したデータと暗号鍵の排他的論理和を行うビット反転データ排他的論理和処理手段を有することは、データ処理と消費電流の関連性を減らすために、有効である。また、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段と入力データのビットを反転したデータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段とを有することは、データ処理と消費電流の関連性を減らすために、有効である。そして、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段と入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段とを有することは、データ処理と消費電流の関連性を減らすために、有効である。

【００１８】それらを組み合わせて、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段と、入力データのビットを反転したデ

ータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段、入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段とを有することは、データ処理と消費電流の関連性を減らすために、有効である。

【００１９】素因数分解が困難であることを利用したRSAなどの暗号では、入力データと暗号鍵を使って、べき乗乗算を繰り返してべき乗剰余計算を行う。データ処理と消費電流の関連性を減らすための方法の一つは、暗号鍵のビットの値に関わらず、入力データとこれまで計算した結果を乗算剰余する乗算剰余処理手段と、暗号鍵のビットが１ならば、乗算剰余処理手段の結果を使い、０であれば、乗算剰余処理手段の結果を無視する乗算剰余処理結果選択処理手段を使用することにより、暗号鍵のビットの値に関わらず、乗算剰余を行うことである。このようにすることにより、乗算剰余のデータ処理と乗算剰余の消費電流の関連性を減り、乗算剰余の実行回数から、暗号鍵を推定することは困難となる。

【００２０】RSAでは、複数ビットの値に対応して入力データのべき乗剰余計算した結果であるビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、暗号鍵の複数ビットの値毎に、ビット値対応入力データべき乗剰余計算処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段を使い、性能を改善する方法がある。しかし、この方法では、常に、同じビット値対応入力データべき乗剰余値の組を使用するため、データと消費電流に関連性から暗号鍵を推定される可能性がある。これを解決する方法一つは、ビット対応乗算剰余計算処理手段を繰り返し処理しているあるタイミングにおいて、ビット値対応入力データべき乗剰余値変更するビット値対応入力データべき乗剰余値変更処理手段を用い、ビット値対応入力データべき乗剰余値を一定の回数で変更する方法である。また、この方法では、変更する方法に依存するが、ビット値対応入力データべき乗剰余値の変更を元に戻す必要がある場合がある。その一つの方法は、ビット値対応入力データべき乗剰余値変更処理手段でビット値対応入力データべき乗剰余値に変更を加えた処理結果を変更を加えなかった値に戻すビット値対応入力データべき乗剰余値変更処理逆変換処理を行う方法である。ビット値対応入力データべき乗剰余値変更処理手段には、いくつかの方法があるが、ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えることも一つの方法である。別な方法は、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると１になる二つの値vとuの中の一方であるv、あるいはvのべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段とビット値対応入力データべき乗剰余値変更処理手段において、vを剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果にuを剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を用いる方法である。また、ビット値対応入力データべき乗剰余値変更処理手段で使う、剰余演算の法Ｎにおいて掛けると１になる二つの値として、２とN+1/2を使うことは、ｖとｕを簡単に求める方法の一つである。RSAのような暗号を処理するために、暗号鍵をＭビットの組として表し、そのビットが０あるいは１であるすべての組み合わせに対して、その値を、入力データにべき乗乗算してビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、あるタイミングで、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると１になる二つの値vとuの中の一方であるv、あるいはvのべき乗を該当剰余演算の法Ｎにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、暗号鍵の複数ビットMの値毎に、ビット値対応入力データべき乗剰余値変更処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手、vを剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果にuを剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を使うことは、データと処理と消費電流の関連性を減らすために有効である。また、その時、あるタイミングで、ビット値対応入力データべき乗剰余値に、vとして２のランダムなべき乗を該当剰余演算の法Ｎにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、２を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に(n+1)/2を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を使うと、ｖとｕを簡単に求めることができる。

【００２１】

【発明の実施の形態】以下、本発明の実施例について図面を参照しながら説明する。

【００２２】図１はICカードの外観を示したものである。ICカード１０１は、IS0７８１６の規格により大きさや、ICカードのチップ１０２の位置や接点の数および割り当てなどが規定されている。

【００２３】図２は、ICカード用チップ１０２の内部構成である。構成については、従来技術の説明で既に述べたとおりである。本発明は、プログラム２０５をいつも一定に実行させるのではなく、ランダムに動作させたり、無駄なダミー処理を追加したりして、ICカード用チ

ップのハードウェアが消費する電流波形の再現を困難にするものである。

【００２４】本実施例では、対称暗号DES(data encryption standard)と非対称暗号RSAを例にする。これは、他の暗号にも使うことができる。DESやRSAなどの暗号については、共立出版株式会社岡本栄司著「暗号理論入門」などに、記載されている。

【００２５】DESは、６４ビットのデータ（平文か暗号文）を５６ビットの暗号鍵で、暗号化と復号化を行う。暗号化と復号化で同じ暗号鍵を使用するため、対称暗号と呼ばれている。DESは、トランプをきって、ランダムにするように、平分（暗号化される文）のビットをランダムに入れ替え暗号化する。データの入れ替えは、暗号鍵にしたがって行う。復号化する時は、暗号鍵に従い入れ替えた順と逆に入れ替えを行い、データを元に戻す。DESの処理での入れ替えは、ビット単位と複数ビットまとまった単位での二つの入れ替え方法がある。前者は、転置と呼び、後者は、換字と呼ぶ。

【００２６】図3,を用いて、DESの暗号化の方法を説明する。暗号文は、まず、初期転置(IP：Initisal Permutation)３０１で転置させられる。これは、初期転置テーブルに従い、ビット単位で、暗号文の６４ビットのデータを入れ替える。これ以降、初期転置の逆転置（IP-1）３１３まで、一組の操作を１６段行う。

【００２７】各１段の処理は、前段の結果の前半か後半の３２ビットのデータと暗号鍵を入力としてｆ関数３０３と呼ばれる処理を行い、その出力を前段の残りの半分のビットを使って排他的論理和３０５を取る操作を行う。暗号鍵も、入れ替えが行われる。暗号鍵に対して、まず、ＰＣ－１というテーブルを使った選択転置PC-1（３０２）が行われる。その後、ＰＣ－２というテーブルを使った選択転置ＰＣ－２　３０４を行い、暗号鍵の入れ替えを行う。次の段では、２８ビットづつをＲＳテーブルにしたがって巡回させて使用する。

【００２８】ｆ関数３０３の処理を、図４に示す。まず、ｆ関数への入力文を選択的転置行列Eに基づいて、選択的転置を行う（４０２）。次に、選択的転置を行った入力データと暗号鍵との排他的論理和をとり（４０３）、Ｓボックスの処理を行い（４０４）、Ｐ転置処理を行う（４０５）。Ｓボックスの処理は、４０３の排他的論理和の処理結果である４８ビットから６ビットづつ取り出し、Ｓボックステーブルの行番号と列番号を求め、４ビットのデータを生成する処理である。各Ｓボックステーブルは、６ビットごとのデータの位置により異なる。Ｐ転置処理は、Ｐ転置テーブルに従い、３２ビットのビット位置を入れ替える操作である。

【００２９】処理手順が同一であれば、データに依存して、消費電流波形の偏りが出るため、本発明では、図４の各処理を、実行ごとに処理手順が変更する、ランダムな処理を追加することを行う。

【００３０】図２４を用いて、選択的転置処理４０２について説明する。選択的転置とは、ｆ関数に入力される入力データを選択的転置行列Eによって、ビットの位置の置き換えと入力データを３２ビットから４８ビットに拡大するものである。選択的転置行列E（２７０１）を図２７に示す。左上（２７０２）から右下（２７０３）に数えて、入力データの第３２、１、２、．．．．、３２、１番目ののビットを出力の第１番目から第４８番目のビットにする。

【００３１】選択的転置処理の手順を説明する前に、本実施例における選択的データ転置で使用するデータについて説明する。ｆ関数に入力されるデータは、図６に示すように、前の処理で正常データ６０２とそのビット６０３を反転したビット反転データの組にして入力される。このような組で扱うと、常に、“０”のビットの数と“１”のビットの数は一定になる。例えば、正常データのビットが、“００１１１１００”であれば、ビット反転データは“１１０００００１１”となり、正常データとそのビット反転データの組での“０”のビットの数は８であり、“１”のビットの数、すなわちハミングウェイトは８である。そのため、正常データとそのビット反転データを組で、データバス（２０３）に転送したならば、組における“０”と“１”のビットの数が常に一定になるため、データの依存性が消費電流に反映されないという特徴がある。例えば、データバス２０３がプリチャージバスであれば、データを転送する準備として、一旦データバスの値を“１”か“０”にそろえ、その後、データを転送する。この場合、常に、プリチャージの状態では、バスの値が同じ値になっているため、正常データとそのビット反転データを組でデータ転送した場合、“１”から“０”あるいは、“０”から“１”に値が変わる数は、プリチャージの状態の前後で、一定になり、データによる消費電流依存性が減る。もし、データバスが、プリチャージ状態を持たない、スタティックバスであれば、ビットの反転は、前のデータの値に依存するが、転送するデータのハミングウェイトが常に一定のため、転送するデータによる電流消費依存性は少ない。少なくとも、正常データだけでは、ハミングウェイトは一定でないため、その電流消費依存性が生じる。正常データとそのビット反転データの組を一緒にデータ転送し、ハミングウェイトをそろえると、転送するデータによる電流消費依存性は減る。

【００３２】選択的転置処理の手順を図２４に示す。

【００３３】選択的転置処理は、入力データの３２ビットを６ビット単位で処理し、４８ビットのデータを生成する。処理されるデータは、正常な入力データとそのビット反転入力データであり、２バイトで実現され、それぞれのバイトに８ビットずつ格納されている。選択的転置処理では、通常、選択的転置行列Ｅの並びの順に処理する。しかし、それでは、データの処理順序が常に一定

であるため、消費電流の波形から、処理されているデータを推測される可能性がある。そのため、本実施例では、実行順序をランダムにし、かつランダムにダミー処理を追加し、処理順序が一定にならないようにする。処理順序がランダムになることにより、データと消費電流の依存性を減らすことができる。

【００３４】本実施例の選択的転置処理では、実行フラグをクリアする（２４０２）。この実行フラグは、各ビットが繰り返し処理の処理単位で処理が終了しているならば、“１”であり、未処理であれば“０”である。実行フラグのすべてのビットが“１”であるならば、処理

実行インデックス＝(乱数 and 0x07) % ８　　（式１）

となる。乱数を0x07で下位３ビットを取り出し、その値を８で割り、その余りを処理する行番号とする。もし、

実行インデックス＝(乱数 and 0x01F) % 32　　（式２）

となる。乱数の下位５ビットを取り出し、その値を３２で割り、その余りを処理するビットの番号とする。実行インデックスは、乱数だけから決まるので、順番はランダムになり、かつすでに処理済みのデータを選ぶこともありえる。本実施例では、データ処理と消費電流の波形の依存関係を減らすために、この性質を積極的に利用し

K ＝ 1を実行インデックス分だけ左にシフトする。　　（式３）

実行フラグ ＝実行フラグ　XOR　K　　（式４）

式３で、実行インデックスのビット位置のビットを“１”にし、式４で、実行フラグと排他的論理和を取ることにより、実行フラグの該当ビットを“１”にする。

【００３８】以上の処理により、繰り返し処理の中の処理の単位が乱数によってランダムに選ばれるため、処理の単位の順序が一定になるない。そのため、データ処理と消費電流の依存性がなくなり、処理や暗号鍵の推測を困難にする。さらに、実行する繰り返しの単位は、ランダムに選ばれるため、同じ処理を１度以上処理する可能性がある。これは、処理時間を不定にするため、データ処理と消費電流の依存性を見出すことをさらに難しくする。

【００３９】次に、実行インデックスで選ばれた繰り返し処理単位での一つの処理について述べる。これは、選択的転置行列Ｅを行ごとの処理の一つ行の処理や選択的転置行列Ｅをビットごとの処理の一つのビットの処理に対応する。まず、実行インデックスに従い、処理すべき正常入力データとビット反転入力データを取り出す（２

アドレス変位 ＝　(乱数AND 0x01)　＊２　　（式５）

これは、乱数の最下位ビットを取り出し、それを２倍する。ルーチンを呼び出すアドレスは、選択的転置ルーチンのアドレスが書かれてあるデータ領域のベースアドレス（２８０２）にアドレス変位を加える。この例では、ベースアドレスには、正常データを最初に処理するルーチンのアドレス（２８０２）が書かれ、ベースアドレス＋２バイトのアドレスには、ビット反転入力データを最初に処理するルーチンのアドレス(２８０３)が書かれて

を終了する（２４０３）。そうでなければ、処理を続け、乱数を取り出す（２４０４）。乱数は、ICカード用チップが乱数生成装置を持っているならば、その装置から乱数を取り出すのも一つの方法である。それ以外の方法として、ソフト的に擬似乱数を作ることも可能である（共立出版株式会社岡本栄司著「暗号理論入門」６１ページから８６ページ）。

【００３５】繰り返し単位でのどの処理を実行するかを乱数によって実行インデックスを決める（２４０５）。例えば、選択的転置行列Ｅを行ごとに処理するのであれば、実行インデックスは、

ビット単位であれば、

ている。

【００３６】次に、実行フラグの対応するビットに処理済みを示すために、“１”にする（２４０６）。一つの手段として、以下の方法がある。

【００３７】

４０７）。本実施例では、正常入力データ以外に反転入力データも同じように選択的転置処理を行う。その時、正常入力データとビット反転入力データがいつも同じ順序で処理されないように、順序をランダム化する。そのために、乱数を取り出し（２４０８）、アドレス変位を計算する（２４０９）。そのアドレス変位から呼び出すルーチンのアドレスを求め（２４１０）、ルーチンを呼び出す（２４１１）。

【００４０】アドレス変位を用いてデータ処理ルーチンを呼び出す一つの実施例を図２８を用いて説明する。メモリ(２８０１)上に、正常入力データを最初に処理するルーチンのアドレス（２８０２）とビット反転入力データを最初に処理するルーチンのアドレス（２８０３）を格納する。二つのアドレスの差がアドレス変位である。アドレスを格納する単位が２バイトとすると、２４０９のアドレス変位の計算方法の一つは以下のようになる。

【００４１】

いる。乱数の最下位ビットが“０”であれば、アドレス変位は０となり、正常入力データを最初に処理するルーチンのアドレス（２８０２）が選択され、正常入力データを最初に処理するルーチン（２８０４）がコールされる。乱数の最下位ビットが“１”であれば、アドレス変位は２となり、ビット反転入力データを最初に処理するルーチンのアドレス（２８０３）が選択され、ビット反転入力データを最初に処理するルーチン（２８０４）が

コールされる。

【００４２】乱数によって、アドレス変位を計算するため、正常入力データを最初に処理するルーチンかビット反転入力データを最初に処理するルーチンがランダムに選択され、正常入力データとビット反転入力データがいつも同じ順序で処理されないようになる。その結果、正常入力データとビット反転入力データの処理と消費電流波形の依存関係がなくなる。

【００４３】正常入力データを最初に処理するルーチンは、図２５に示すように、正常入力データの選択的転置を行い（２５０２）、次に、ビット反転入力データの選択的転置を行う（２５０３）。ビット反転入力データを最初に処理するルーチンは、図２６に示すように、ビット反転入力データの選択的転置を行い（２６０２）、次に、正常入力データの選択的転置を行う（２６０３）。実際の選択的転置を行う処理ルーチン（２８０６）は、それぞれのルーチンからさらに呼び出され、対応するビットの選択的転置が行われる。そして、ビット反転入力データの選択的転置（２５０３）あるいは（２６０２）では、正常データを選択的転置を行った結果のビット反転したデータを生成する。これは、選択的転置がビット位置の転置であるため、正常データとビット反転データのビット反転の関係は維持されるためである。正常データとビット反転データの格納方法は、図６で示したように、組みになるように格納する。こうすることにより、データ転送時のデータと消費電流の関連性を少なくすることができる。

【００４４】以上の処理により、繰り返し処理すべきデータの一つの処理が終わると、２４０３に戻る。すべてのデータを処理していなければ、乱数に基づき別の処理すべきデータをランダムに選び、処理を繰り返す。乱数を使ってのみ処理すべきデータを決めているので、同じデータを処理することもある。選択的転置処理では、同じデータに対して２度以上を繰り返しても問題が起きない。

【００４５】以上説明したように、本実施例の選択的転置処理において説明した、正常入力データとビット反転入力データは、請求項１の「正常データとそのビット反転データの組」の実施例である。また、図２５の正常入力データを最初に処理するルーチンあるいは図２６のビット反転入力データを最初に処理するルーチンは、ルーチンの中に、正常データの処理手順とビット反転の処理手順を持っており、請求項３の「正常データ命令及びビット反転データ命令処理手段」の実施例である。また、図２５の正常入力データを最初に処理するルーチンあるいは図２６のビット反転入力データを最初に処理するルーチンは、正常入力データの選択的転置を行った結果を作成すると同時に、その結果をビット反転した結果も生成する。これは、請求項４の「ビット反転データ生成手段」の実施例の一つでもある。データ処理ルーチン２８０６は、選択的転置処理では、実際に、その処理を行うルーチンであり、正常データでもビット反転データでも使用されるルーチンである。これは、請求項２の「正常データ及びビット反転データ処理手段」の一つの実施例である。また、実行フラグと乱数により処理順序をランダムに変える手順（２４０２から２４０５）は、請求項５の「繰り返しランダム実行処理手段」の一つの実施例である。実行フラグと乱数によりすでに処理済みのデータもランダムに２度以上処理を行う手順（２４０２から２４０６）は、請求項６の「ダミー実行処理手段」の一つの実施例である。そして、実行フラグと乱数を使った処理手順２４０２から２４０６は、ダミー処理と繰り返しランダム処理を同時に実現しているため、請求項７の「ダミー実行及び繰り返しランダム実行処理手段」の実施例の一つでもある。図２４で述べた選択的転置で述べた処理は、請求項８の典型的な実施例である。

【００４６】次に、選択転的置を行った入力データと暗号鍵との排他的論理和処理（４０３）の実施例について述べる。この処理の基本的な流れは、図２４を用いてすでに説明した選択的転置の処理の手順とほとんど同じである。主として異なっている点は、７１０での呼び出しアドレスの計算が排他的論理和のルーチンのベースアドレスに基づいていることである。排他的論理和処理は、選択的転置処理で作成された、二つのデータを処理する。一つは、正常入力データを選択的転置した正常E転置データであり、もう一つは、ビット反転入力データを選択的転置したビット反転E転置データである。これらは、図６に示した形式で格納し、取り出すときは、正常データとビット反転データの対を一度に取り出す。ビット反転E転置データは正常E転置データのビット反転となっている。正常E転置データと暗号鍵は、ともに４８ビットである。また、次の処理であるＳボックス処理が６ビット単位の処理であるため、排他的論理和処理では６ビット単位の処理を行うことが多い、１ビット単位で排他的論理和処理の実行も可能である。前者では、少なくとも８回の繰り返し処理（７０３から７１１）が必要であり、後者では、少なくとも４８回の繰り返し処理（７０３から７１１）が必要である。

【００４７】正常E転置サブデータを最初に処理するルーチンでは、正常E転置サブデータと暗号サブ鍵の排他的論理和を最初に実行し（８０２）し、ビット反転E転置サブデータと暗号サブ鍵の排他的論理和を後に実行する（８０３）。ビット反転E転置サブデータを最初に処理するルーチンでは、実行順序が逆転する（９０２と９０３）。

【００４８】正常E転置サブデータと暗号サブ鍵の排他的論理和の結果とビット反転E転置サブデータと暗号サブ鍵の排他的論理和は、ビット反転の関係になっている。これは、正常データとビット反転データを同じ暗号サブ鍵と排他的論理和を実行したため、ビット反転の関

係が保存されているためである。

【００４９】正常E転置サブデータを最初に処理するルーチン（８０１）とビット反転E転置サブデータを最初に処理するルーチン（９０１）は、正常データと暗号鍵データの排他的論理和とビット反転データの排他的論理和を行うため、請求項１０の実施例である。

【００５０】次に、Ｓボックスの処理４０４について述べる。本実施例でのＳボックスの処理も基本的な流れは、図２４の選択的転置の処理と似た流れである。

【００５１】Ｓボックスの最初に、Ｓボックスの処理の開始をランダムにするために、乱数を取り出し（１００２）、乱数値に従いダミー処理を実行する（１００３）。このダミー処理は、ループ処理がないループを乱数値だけ繰り返すような処理でもよい。乱数値に従い、ループの回数が異なるため、Ｓボックスの処理の開始がランダムになり、データ処理と消費電流の波形の依存性が少なくなる。

【００５２】実行フラグと乱数を使った処理順序のランダム化とダミー処理を追加する処理（１００４から１０１３）は、選択的転置処理の２４０２から２４１１の処理と同じである。Ｓボックス処理は、Ｅ転置データと暗号鍵との排他的論理和の結果を使って行う。暗号鍵と選択的転置の実行結果の排他的論理和処理（４０３）の実行結果も図６のような正常データとビット反転データの対の配列からなるデータである。実行インデックスを持つ排他的論理和の実行結果を取り出すときは（１００９）、同様に対のデータを取り出す。正常排他的論理和の実行結果を最初に処理するルーチン（１１０１）とビット反転排他的論理和の実行結果を最初に処理するルーチン（１２０１）は乱数でランダムに呼ばれる。

【００５３】Ｓボックスの処理は、排他的論理和の実行結果の６ビット毎に、Ｓボックステーブルのアドレスを計算して（１７０２）、Ｓボックスの変換結果である４ビットのデータを求める（１７０３）。アドレスの計算は、６ビットの第１ビットと第６ビットの２ビットで行番号を求め、第２ビットから第５ビットの４ビットで列番号を求める。

【００５４】Ｓボックステーブルは、８個ある。それぞれのＳボックステーブルは、排他的論理和を処理した結果の６ビットずつの変換に使用する。通常の１番目のＳボックステーブル（２９０１）の１列目から４列目、１行目から４行目のデータを図２９に示す。本実施例では、図３０のようにＳボックステーブルをビット反転データも求めるように拡張する。図３０での表示範囲は、同じであるが、Ｓボックステーブルの要素を二つのフィールドに拡張する。第１のフィールド(列の左側のフィールド、例えば３００４)は、図２９のＳボックステーブルの要素のデータを格納し、第２のフィールド(列の右側のフィールド、例えば３００５)は、第１のフィールドのデータのビット反転データを格納する。

【００５５】正常排他的論理和の実行結果を最初に処理するルーチン（１１０１）では、まず、正常な排他的論理和の実行結果から、拡張したＳボックステーブルを使い、正常Ｓボックス処理データとビット反転Ｓボックス処理データを求める（１１０２）。次に、ビット反転排他的論理和の実行結果から、拡張したＳボックステーブルを使い、正常Ｓボックス処理データとビット反転Ｓボックス処理データを求める（１１０３）。ビット反転排他的論理和の実行結果を使って、Ｓボックス処理をしているため、１１０３の処理結果は、使用しない。しかし、データ処理と消費電流の関連性を減らすために、ダミー処理として実行する。ビット反転排他的論理和の実行結果を最初に処理するルーチン（１２０１）では、処理順序が逆になり、同様に、ビット反転排他的論理和の実行結果からの正常Ｓボックス処理データとビット反転Ｓボックス処理データは、使用されない。次のＰ転置処理に渡されるデータは、正常排他的論理和の実行結果を使った正常Ｓボックス処理データとビット反転Ｓボックス処理データである。

【００５６】排他的論理和の実行結果から拡張したＳボックステーブルを使って、正常Ｓボックス処理データとビット反転Ｓボックス処理データを求めるには、まず、排他的論理和の実行結果の６ビットからＳボックステーブルのアドレスを計算する（１７０２）。次に、拡張したＳボックステーブル（３００１）の正常Ｓボックス処理データを第１のフィールドから求め、ビット反転Ｓボックス処理データを第２のフィールドから求める（１７０３）。

【００５７】Ｓボックスの処理での正常Ｓボックス処理データとビット反転Ｓボックス処理データは、請求項１の「正常データ及びビット反転データの組み」の実施例の一つである。また、拡張Ｓボックステーブル（３００１）を使って、正常Ｓボックス処理データとビット反転Ｓボックス処理データを生成する処理（１７０３）は、請求項４の「ビット反転データ生成手段」の実施例でもある。ビット反転排他的論理和の実行結果から、正常Ｓボックス処理データとビット反転Ｓボックス処理データを求めるＳボックステーブルを使いＳボックス変換結果を計算する処理（１１０３と１２０２）は、正常データではなく、ビット反転データを処理することにより、正常データもビット反転データも同じように処理をすることにより、データ処理と消費電流の依存性を減らすものである。これは、請求項３の「正常データ命令及びビット反転データ命令処理手段」の実施例の一つである。Ｓボックスの処理は、６ビット単位のビット単位ではない換字処理であるので、請求項９の実施例である。正常排他的論理和の実行結果を最初に処理するルーチン（１１０１）あるいは、ビット反転排他的論理和の実行結果を最初に処理するルーチン（１２０１）は、正常データに対して換字を行い、換字の結果とそのビット反転した結

果を生成し、かつビット反転データに対して換字を行い、換字の結果とそのビット反転した結果を生成するため、請求項１１の実施例である。

【００５８】Ｐ転置処理(４０５)について図１３を用い説明する。Ｐ転置処理も選択的転置処理（２４０１）と同じである。異なっている点は、Ｐ転置行列（図１６の１６０１）を使用していることである。

【００５９】Ｓボックス処理からは、図６のデータ形式で、正常Ｓボックス処理データとビット反転Ｓボックス処理データが渡される。正常Ｓボックス処理データを最初に処理するルーチン（１４０１）では、正常Ｓボックス処理データのＰ転置を行い（１４０２）、次にビット反転Ｓボックス処理データのＰ転置を行う（１４０３）。ビット反転Ｓボックス処理データを最初に処理するルーチン(１５０２)では、順序が逆になっている。Ｐ転置は、ビットの位置の入れ替えのため、ビット反転Ｓボックス処理データの処理結果であるビット反転Ｐ転置処理データは、正常Ｓボックス処理データの処理結果である正常Ｐ転置処理データのビット反転の関係になっている。１４０１と１５０１は、請求項１２の実施例である。

【００６０】以上、ｆ関数の処理（４０２から４０５）の実施例を示した。これは、請求項１３の実施例である。本実施例では、データ処理の順番のランダム化、ダミー処理の追加、正常データやビット反転データを使うことにより、データ処理とICカードチップが消費する電流との関連性をわかりにくくし、消費電流の波形を観測しても暗号鍵の推定を困難にすることができる。

【００６１】次に、RSAのための実施例を説明する。RSAは、素因数分解が困難なことを利用した暗号アルゴリズムである。暗号化及び復号化では、異なった鍵を使うためを、非対称暗号アルゴリズムと呼ばれる。暗号化も復号化も以下のべき乗剰余の計算を使用する。

【００６２】

y＝x ** e mod n （式６）

a = b mod nは、a - bがnで割り切れることを示す。復号化のときは、xは、暗号文で、eは秘密鍵であり、yが複号化された平文である。べき乗剰余の計算は、乗算剰余を使って計算することができる。ICカードでは、べき乗剰余を高速に計算できるために、乗算乗除を計算できるコ・プロセッサを持っていることがある。乗算剰余は、以下の式である。

【００６３】y = a * b mod n （式７）

べき乗剰余を乗算剰余を使って計算する方法を図１８に示す。この計算方法は、例えば、共立出版株式会社岡本栄司著「暗号理論入門」の９４ページに示されている。秘密鍵である整数eの２進数展開をe = e0 e1 e2 ..... e (w-1) とする。すなわち、２進数展開した結果（今後、eのビットという）は、eを２進数で表したビット列になる。yの初期値を１とし（１８０２）、eの最上位のビットから、yをnの法で二乗剰余し（１８０５）、該当ビットの値が“１”であれば、yにnの法でxを乗算剰余する（１８０７）。“０”であれば、１８０７の処理は、スキップする。１８０５から１８０７の処理を、eの最上位のビットから最下位のビットまで計算する。

【００６４】図１８の処理手順では、eのビットが“１”であれば、コ・プロセッサが２回動き、“０”であれば、１回しか動かないため、コ・プロセッサの動きを観測すれば、秘密鍵のeを推定することができる。eを２進数展開したビットに関わらず、コ・プロセッサの動作回数を同じにすると、コ・プロセッサの動きを観測するだけでは、秘密鍵を推定することが困難になる。

【００６５】図１９に、コ・プロセッサの動作回数を同じにして、秘密鍵を推定することが困難にしたべき乗剰余の処理手順を示す。本実施例では、秘密鍵eのビットの値に関わらず、yにxを乗算剰余するする（１９０６）。その代わりeのビットが“１”であれば、その結果をyにいれる（１９０８）。そうでなければ、ｙをそのまま使用する（１９０９）。これをeの最下位ビットまで処理を続ける。本実施例では、eのビットの値に関わらず、コ・プロセッサの動作回数が同じであるため、消費電流波形からコ・プロセッサの動きを観測しても、秘密鍵eのビットの値は分からない。本実施例は、請求項１４の実施例である。請求項１４の乗算剰余処理手段は１９０６として実現されており、乗算剰余処理結果選択処理手段は１９０７から１９０９によって実現されている。

【００６６】図１９に示したべき乗剰余計算を計算する方法より高速な方法として、図２０に示す方法がある。共立出版株式会社岡本栄司著「暗号理論入門」の９５ページにて記載されているアディション・チェインという方法を使ったものである。これは、図１９に示した方法が、eの１ビットずつの処理であったのに対し、複数ビットをまとめて処理する方法である。図２０に示す方法は、２ビットをまとめて示す方法である。まず、べき乗算される暗号文xの０乗、１乗、２乗、３乗した結果を求める（２００２）。次に、yを１に初期化し（２００３）、そして、最後のビットまで処理していなければ（２００４）、yを２回二乗剰余を行う。これは、yの値を２ビット分上位の桁にシフトすることに対応する。そして、eの上位の２ビットずつから取り出して、それらが“００”、“０１”、“１０”、“１１”の値にしたがって、２０１０、２０１１、２０１２、２０１３の処理を行う。それぞれの処理では、ｅの２ビットの値によって、べき乗剰余計算の途中結果が格納されているyにt[0]からt[3]までの該当する値が乗算剰余される。例えば、２０１２の処理は、ｅのビットが“１０”の場合で、ｙにｘの二乗剰余の結果であるt[2]を掛ける。その後、最後のビットを処理するまで、２００４から２０１

３までの処理を繰り返す。なお、最後にｅが１ビットだけ残った場合は、その１ビット分は図１８か図１９の処理を行う。

【００６７】図２０では、eの２ビットがどの値でも、コ・プロセッサーは、２００５で2回、２０１０から２０１３のどれか1回を実行するため、合計同じ３回動作する。そのため、コ・プロセッサーの動作を観測するだけでは、ｅのビットを推定することは困難である。なお、図１９の処理では、１ビットの処理をするために、２回コ・プロセッサーが動いたが、図２０の処理では、２ビットを処理するために、３回コ・プロセッサーが動作しており、１ビットあたり１．５回コ・プロセッサーが動作する。そのため、図２０の２ビット方式の方が図１９の１ビットの方式に比べて、約２５％高速である。ここで、処理時間がコ・プロセッサーの動作回数に比例する理由は、図２０も図１９の処理の大部分がコ・プロセッサーによる演算で処理時間を消費しているためである。

【００６８】しかし、ステップ２０１０からステップ２０１３でのt[0]からt[3]は、ｅの２ビットごとの処理で毎回同じ値を使用するため、データと消費電流の波形の関連が分かれば、ｅのビットを推定される恐れがある。

【００６９】本発明では、t[0]からt[3]までの値を、べき乗剰余演算中に変更することにより、データと消費電流の波形の関連性を減らすものである。図２１を用いて、実施例の一つを説明する。図２１と図２０の違いは、図２０の処理の流れに２１０５と２１０６が追加されたことである。図２１の実施例は、べき乗剰余を行うためにｅを２ビットずつ繰り返し処理中に、ある回数毎に、t[i]までの値を変更する。２１０５では、繰り返しが一定の回数を処理したかを調べる。そうであれば、２１０６でｔ[i]の値を変更する。２１０５での一定の回数として、ｅの桁を２０で割った値が考えられる。これは、２ビット処理を行っているため、t[i]の値を１０回変更することに対応する。また、任意の桁を超えたときにｔの値を変更するようにも可能である。t[i]の値の変更方法は、いろいろな方法がある。本発明の本質は、べき乗剰余を行うための繰り返しの中でt[i]の値を変更することである。２１０６では、t[i]に、剰余演算の法であるｎに乱数を掛けて加えることにより、t[i]の値を変更する。これは、以下のように行う。

【００７０】

r = 乱数　（式８）

t[i] =t[i] + r * n　（式９）

変更されたt[i]は、２１１２から２１１５のどれかの処

t[i] = t[i] * v mod n

そして、べき乗剰余が終了したときに、掛けた値のｎの法での逆数を掛け合わせた分だけ掛けることにより元に戻す方法である。

【００７５】

理で、以下のようにyとｎを法として乗算剰余される。

【００７１】

y = y * t[i] mod n　（式１０）

= y * (t[i] + r * n) mod n

= y * t[i] mod n + r * n mod n

= y * t[i] mod n

この処理で、ｒ＊ｎの影響は計算から消えてなくなる。これは、ｒ＊ｎ mod nがゼロであるためである。すなわち、r*nはｎで割り切れるためである。t[i]は、t[i]と異なった値のため、ある回数毎に、２１１２から２１１５で使われているt[i]が変わるため、データと消費電流の波形の関連性を減らすことができる。本実施例は請求項１５の実施例である。実施例１５の「ビット値対応入力データべき乗剰余計算処理手段」は２１０２であり、「ビット対応乗算剰余計算処理手段」は、２１０８から２１１５である。「ビット値対応入力データべき乗剰余値変更処理手段」は、２１０５と２１０６で実現されている。また、本実施例は、請求項１７の実施例でもある。実施例１７の「ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えるビット値対応入力データべき乗剰余値変更処理手段」は、２１０６として実現されている。

【００７２】「ビット値対応入力データべき乗剰余値変更処理手段」としての別な実施例として、剰余演算の法に掛ける値として２の任意の数ｒのべき乗の値を使用する方法がある。２のべき乗を使うことにより、法Ｎに２のｒ乗を掛ける処理が容易になる。法Ｎに２のｒ乗を掛けることは、法Ｎをｒビット分左にシフトするだけで２のｒ乗を掛けること等価になる。一般に法Ｎは１００ビットから１０００ビットまでの整数であるため、法Ｎと通常の乱数ｒの掛け算は、処理時間がかかる。シフト操作は、掛け算に比べてきわめて高速である。また、２のべき乗ではなく、２の８乗のべき乗を使うと、シフトが１ビットシフトではなく、８ビットシフト、すなわち１バイトシフトとなり、ビットのシフトより更に高速に実現できる。１バイト左へのシフトは、データを１バイト分左に置くことに相当し、シフト処理ではなく、データ移動命令で実現できるからである。本実施例は、請求項２２の実施例である。

【００７３】t[i]の値の変更方法として、別の実施例を図２２を用いて説明する。図２２での変更方法は、t[i]を変更するときに、以下の式のようにある値を乗算剰余して変更する。

【００７４】

（式１１）

k = vをべき乗剰余した回数　（式１２）

s = u ** k mod n　（式１３）

y = y * s mod n　（式１４）

ここで、u * v mod n = 1

(式１５)

である。これは、以下の式に示すように、途中でvを掛けた回数だけuを掛けて、元に戻すという方法である。

【００７６】

```
y = y * v**k mod n                                  (式１６)
y = y * u**k mod n                                  (式１７)
= y * v**k * u**k mod n
= y * (v*u)**k mod n
= y mod n * (v*u)**k mod n
= y mod n * ((v*u) mod n)**k mod n
= y mod n * 1 mod n
= y mod n * 1
= y mod n
```

yは、途中でvを掛けているため、yがｋでvをべき乗剰余された値である。yは、yにｕをｋでべき乗剰余したものを掛けたものであり、式１５を使うことにより、結果はy mod nとなる。

【００７７】処理の流れを図２２で説明する。まず、２２０４で法ｎの剰余演算でvとその逆数であるuを求める。そして、２２０７で、ｖが一定にならないように、ｖを変形する。変形の一つ方法は、乱数を求め、その乱数でvをべき乗剰余する。また、一定回数毎に、ｖをべき乗する数を増やすことも一つの方法である。その値をt[i]に乗算剰余する（２２０８）する。そして、ｅのビットの処理が終わったならば、ｕを使って、ｖの操作を無効にする演算を行い、正しいｙの値を求める（２２１８）。ｎがあらかじめわかっており、かつあらかじめ定められた乱数を使った場合、ｕをｖを掛けた回数だけ掛けた結果ｓをあらかじめ求めておくことができる。その値

```
t[i] = t[i] * v mod n
y = y * t[i] mod n
  = y * t[i] * v mod n
y = y * u mod n
  = y * t[i] * v * u mod n
  = y * t [i] mod n
```

2308で式２０のようにt[i]は変形されており、２３１４から２３１７のどれかでは式２１が実行される。その結果を使い、２３１８で式２２の処理を行い、元の式の結果を導出する。

【００８１】さらに効率的に実行するためには、毎回、vを掛けたことを戻すのではなく、一定回数ごとにvを掛けたことを戻してもよい。その時は、式１９のuを一定回数ｖを掛けたことを戻す値を選び、式１９を一定回数毎とに計算するようにすればよい。

【００８２】図２３において、ｖとして(n+1)/2を用い、uとして２を用いる。vとuは、ｎを法とする剰余演算で逆数の関係にある。すなわち、ｖ＊ｕ mod ｎ＝１である。２３０７のｖの変形処理として、最初の一定回数の繰り返し処理では、

ｓを使って、

```
y = y * s mod n                                     (式１８)
```

とし、ｙの元の値を計算することができる。これは、式１７の操作と同じである。

【００７８】本実施例は、請求項１８の実施例である。請求項１８の「ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると１になる二つの値vとuの中の一方であるv、あるいはvのべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段」は２２０４と２２０７、２２０８で実現されている。また、本実施例は、請求項１６の実施例でもある。請求項１６の「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」の実施例は、２２１８である。

【００７９】次に、図２３の処理で、「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」をｅのビットをすべて計算した後ではなく、ｅのビットの計算途中に行う実施例を示す。図２３の処理では、２３０２から２３１７までの処理は、図２２の２２０２から２２１７までの処理と同じである。２２１８に相当する処理を（２３１８）、２３１４から２３１７のどれかを実行した後に行う。例えば、２３０８でt[i]にvを乗算剰余した場合、その操作を毎回元に戻す場合は、ｙ＝ｙ＊ｕ mod ｎ　　(式１９)を行えばよい。これは、２３１４から２３１７でyに掛けられているt[i]は、元にt[i]のvを掛けたものであるからである。

【００８０】

(式２０)

(式２１)

(式２２)

```
v = 1                                               (式２３)
```

それ以降の繰り返し処理では、

```
v = v * (n+1)/2                                     (式２４)
```

を用いる。

【００８３】そして、２３１８で、vが(n+1)/2であるときの計算を３回毎に戻すならば、

```
u = ((2**4*2)**4*2)**4*2                            (式２５)
```

となり、2**85となる。式２５は、図２３の処理では、２ビット処理を使っているため、ｖをｙに掛けるのは２ビットごとにしか行われないためと、直前に２３１４から２３１７でｖを掛けていることを元に戻すための式である。そして、２３１８では、３回毎に

```
y = y * 2**85 mod n                                 (式２６)
```

を計算すれば、 t[i]に(n+1)/2を掛けた操作は元に戻

る。２３０７で、ｖが式２４を使って、((n+1)/2)**2と

u = (((2**2)**4*2**2)**4*2**2)**4*2**2

となり、(2**2)**85 = 2*170となる。そして、２３１８では、３回毎に

y = y * 2**170 mod n （式２８）

を計算すれば、t[i]に((n+1)/2)**2を掛けた操作は元に戻る。式２５や式２７は、vとして(n+1)/2を使うため、あらかじめ値を計算しておく事ができる。図２３では、ｅを２ビットずつ処理する方式で述べたが、図２１や図２２、図２３で述べた方式は、eを１ビットずつ処理する方式や３ビットあるいは４ビットごとに処理する方式にも適用できる。

【００８４】図２３の２３１８は、請求項１６の「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」を、ｅのビットをすべて処理した後まとめて実行するのではなく、途中で実行する実施例である。v=(n+1)/2とu=2を使った式２１から式２８の実施例は、請求項１９の実施例である。図２２及び図２３の処理は、請求項２０の実施例である。請求項２０の「ビット値対応入力データべき乗剰余計算処理手段」は、２２０２あるいは２３０２の処理である。請求項２０の「ビット値対応入力データべき乗剰余値変更処理手段」は、２２０７と２２０８、あるいは２３０７と２３０８で実現されている。請求項２０の「ビット対応乗算剰余計算処理手段」は２２１０から２２１７、あるいは２３１０から２３１７で実現されている。請求項２０の「入力データべき乗剰余値変更逆変換処理手段」は２２１８あるいは２３１８で実現されている。

【００８５】図２１から図２３で述べた実施例は、ＲＳＡに基づいた実施例であるが、剰余演算を使用したその他の暗号、たとえばエルガマル暗号にも同様に適用できる。

y * r = (a * r) mod (r * n)

となる。すなわち、法Ｎをｒ倍し、右辺のaをｒ倍すると、左辺もｒ倍となることを示している。法Ｎのデータと消費電流の関連性を消すためには、法Ｎを変形することが一つの方法である。式３２から、法Ｎにｒを掛けて、かつ右辺のaをｒ倍して、剰余演算をし、その結果をｒで割れば、もとのｙを求めることができることを示している。このようにすることにより、法Ｎの値と消費電流の関連性を隠すことができる。これは、特に「中国人の剰余定理」（共立出版株式会社岡本栄司著「暗号理論入門」の９６ページ）を使ってＲＳＡの計算を行うときに、法Ｎの素因数ｐとｑを使った剰余計算においてｐとｑの情報を隠蔽するために有効である。

【００９１】図３１でその処理手順を示す。まず、法Ｎを変形するために法Ｎに掛ける値ｒを求める（３１０２）。次に、法Ｎを変形するためにＮをｒ倍する（３１０３）。そして、右辺の値もｒ倍し（３１０４）、剰余演算を行う（３１０５）。そして、結果を元に戻すために、演算結果をｒで割る（３１０６）。

変形されたならば、式２５は、

（式２７）

る。図２１から図２３に示した「ビット値対応入力データべき乗剰余計算処理手段」、「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」は、より一般的な剰余演算にも適用できるため、請求項２６から請求項３３までの実施例でもある。また、請求項１５から請求項２２までは、共立出版株式会社岡本栄司著「暗号理論入門」の９５ページにて記載されているアディション・チェインという方法をベースにしたが、通常の１ビットごとの乗算剰余を行うべき乗剰余演算にもでも使うことができる。

【００８６】剰余演算でのデータと消費電流波形の関連性は、乗算剰余演算の乗数と被乗数だけではなく、法Ｎの値にも依存している。法Ｎの値と消費電流波形の関連性を隠すことが、重要な情報を外部に漏らさないために必要となる。ＲＳＡなどの暗号処理では、以下のような計算が頻繁に出てくる。

【００８７】

y = a * b mod n （式２８）

これを一般化すると、以下の剰余計算となる。

【００８８】

y = a mod n （式２９）

これは、aをnで割った余りがyであることを示しており、以下の等式と等価である。

【００８９】

a = n * x + y （式３０）

この両辺に任意の数ｒを掛けると、以下のようになる。

【００９０】

a * r = n * x * r + y * r （式３１）

これを剰余計算の式であらわすと

（式３２）

【００９２】図３１の３１０３は、請求項２３の剰余演算の法を変更する剰余演算法変更処理手段であり、３１０４は、剰余演算の右辺の値を変更する剰余演算右辺変更処理手段であり、３１０６は、剰余演算法変更処理手段で変更した剰余演算の法を変更前の計算結果に戻す剰余演算法変更処理逆変換処理手段の実施例である。図３１では、請求項３２に示すように剰余演算の法と剰余演算の右辺を変更する手段として、任意の数ｒを掛けている。そして、剰余演算法変更処理逆変換処理手段では、ｒで割ることを使用している。

【００９３】法Ｎなどを変形するｒとして、２のべき乗を用いると、法Ｎと右辺の値のｒ倍の処理は、べき乗分Ｎを左にシフトすればよい。そして、剰余演算の結果をｒで割る処理はその値を右にシフトするだけでよく、掛け算や割り算が出てこないため、高速に演算ができる。また、２の８乗のべき乗をｒとして用いると、掛け算や割り算は、シフト命令ではなく、データ移動命令となり、より高速に実現できる。本実施例は、請求項２５の

実施例である。

【００９４】

【発明の効果】本発明によれば、ICカードチップでの処理手順をランダムにすることや、ダミー処理を挿入すること、処理データを変形することにより、消費電流の波形から、処理や暗号鍵の推測を困難になる。

【図面の簡単な説明】

【図１】ICカードのハードウェア構成。

【図２】ICカード用チップ内のハードウェア構成。

【図３】DESの全体処理の流れ。

【図４】DESのｆ関数の処理の流れ。

【図５】消費電流の波形。

【図６】正常データとビット反転データのデータ構造。

【図７】暗号鍵と選択的転置の結果と排他的論理和の処理。

【図８】正常E転置サブデータを最初に処理するルーチン。

【図９】ビット反転E転置サブデータを最初に処理するルーチン。

【図１０】Ｓボックス処理。

【図１１】正常排他的論理和の実行結果を最初に処理するルーチン。

【図１２】ビット反転排他的論理和の実行結果を最初に処理するルーチン。

【図１３】Ｐ転置処理。

【図１４】正常Ｓボックス処理データを最初に処理するルーチン。

【図１５】ビット反転Ｓボックス処理データを最初に処理するルーチン。

【図１６】Ｐ転置行列。

【図１７】Ｓボックステーブルを使いＳボックス変換処理するルーチン。

【図１８】RSAの処理ルーチン。

【図１９】暗号鍵のビットの値に関わらず乗算剰余処理を行うように変更したRSAの処理ルーチン。

【図２０】２ビット方式のRSAの処理ルーチン。

【図２１】剰余演算の法の整数倍を加えることによりビット値対応入力データべき乗剰余値を変更するRSAの処理ルーチン。

【図２２】ビット値対応入力データべき乗剰余値変更処理逆変換処理を最期に実行するRSAの処理ルーチン。

【図２３】ビット値対応入力データべき乗剰余値変更処理逆変換処理を途中で実行するRSAの処理ルーチン。

【図２４】選択的転置処理。

【図２５】正常入力データを最初に処理するルーチン。

【図２６】正常入力データを最初に処理するルーチン

【図２７】選択的転置行列Ｅ。

【図２８】アドレスを使ったデータ処理ルーチンの呼び出し。

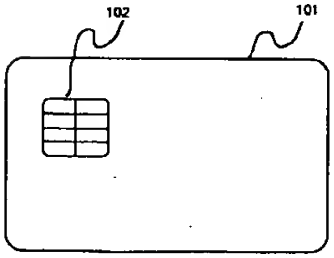
【図２９】Ｓボックステーブル。

【図３０】拡張したＳボックステーブル。

【図３１】剰余演算の法を変形する処理手順。

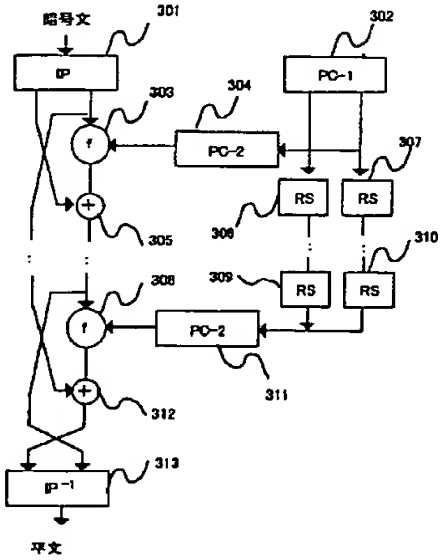
【図１】

図1



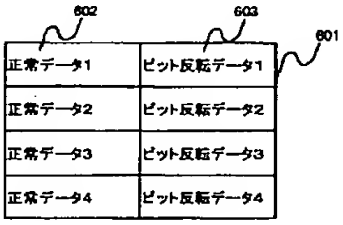
【図３】

図3



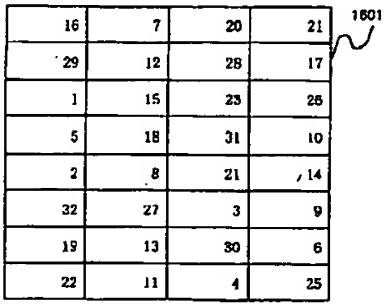
【図６】

図6



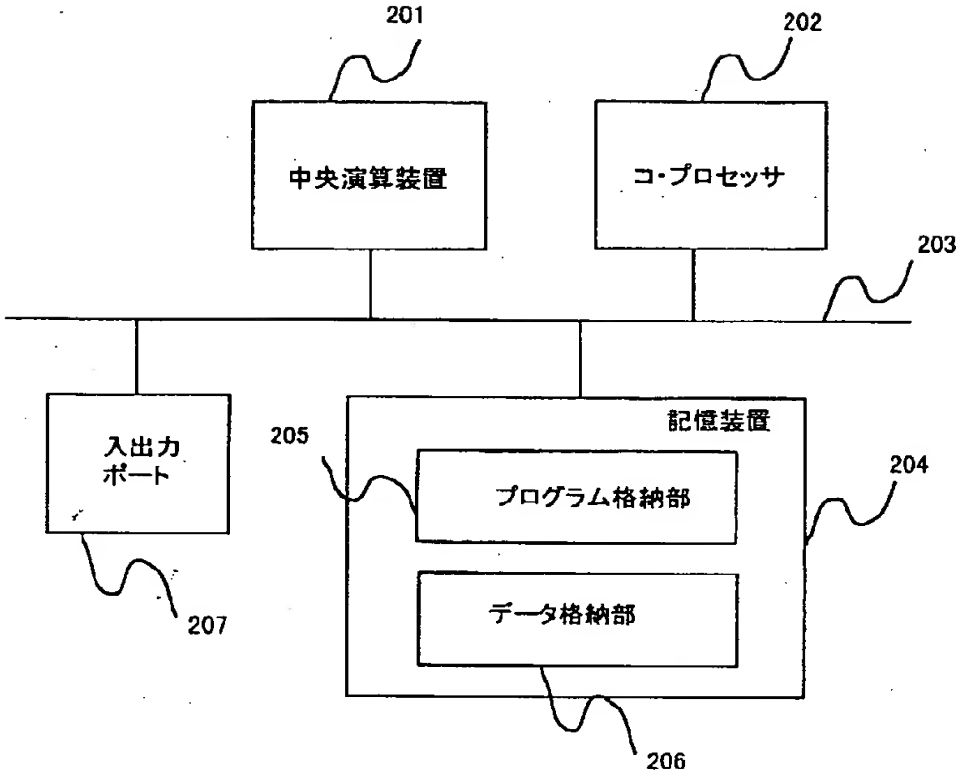
【図１６】

図16



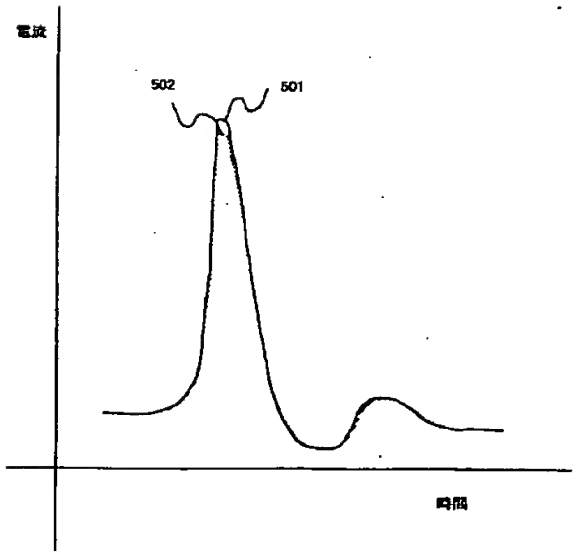
【図２】

図2



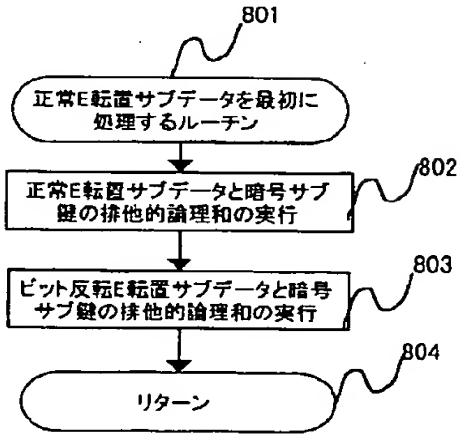
【図５】

図5



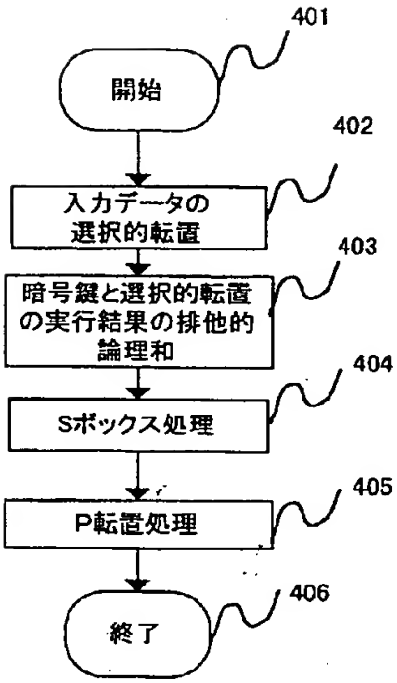
【図８】

図8



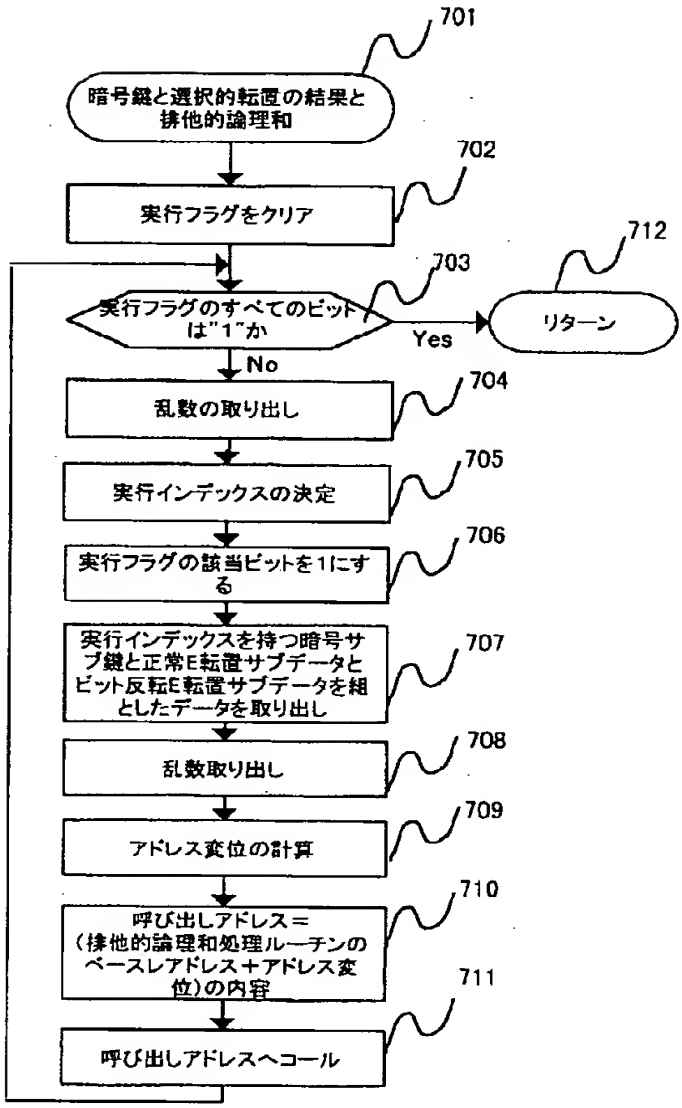
【図４】

図4


401
開始
402
入力データの選択的転置
403
暗号鍵と選択的転置の実行結果の排他的論理和
404
Sボックス処理
405
P転置処理
406
終了

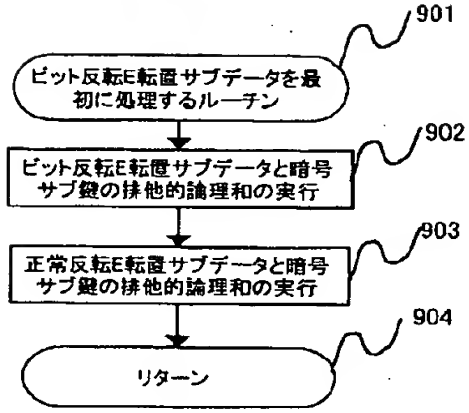
【図７】

図7


701
暗号鍵と選択的転置の結果と排他的論理和
702
実行フラグをクリア
703
実行フラグのすべてのビットは"1"か
Yes
712
リターン
No
704
乱数の取り出し
705
実行インデックスの決定
706
実行フラグの該当ビットを1にする
707
実行インデックスを持つ暗号サブ鍵と正常E転置サブデータとビット反転E転置サブデータを組としたデータを取り出し
708
乱数取り出し
709
アドレス変位の計算
710
呼び出しアドレス＝（排他的論理和処理ルーチンのベースレアドレス＋アドレス変位）の内容
711
呼び出しアドレスへコール

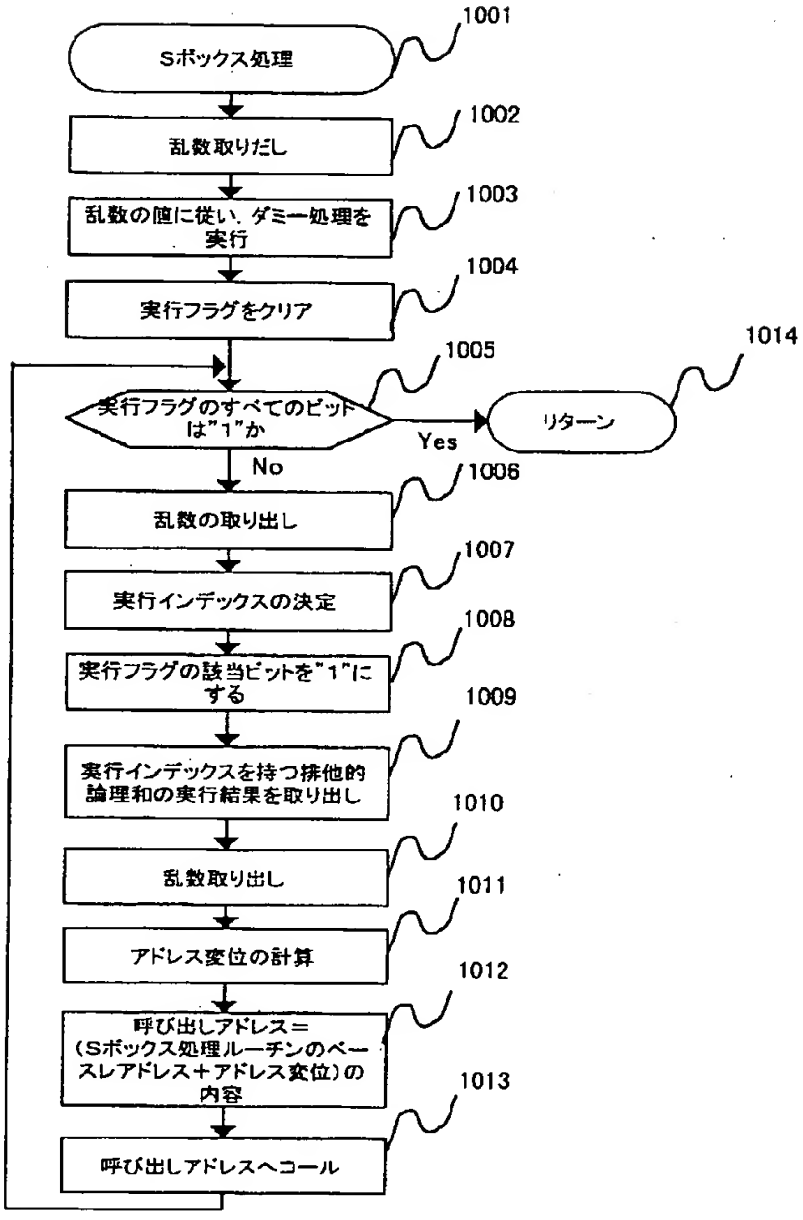
【図９】

図9



【図１０】

図10



【図２７】

図27

2701		2702				
	1	2	3	4	5	6
1	32	1	2	3	4	5
2	4	5	6	7	8	9
3	8	9	10	11	12	13
4	12	13	14	15	16	17
5	16	17	18	19	20	21
6	20	21	22	23	24	25
7	24	25	26	27	28	29
8	28	29	30	31	32	1

2703

【図２９】

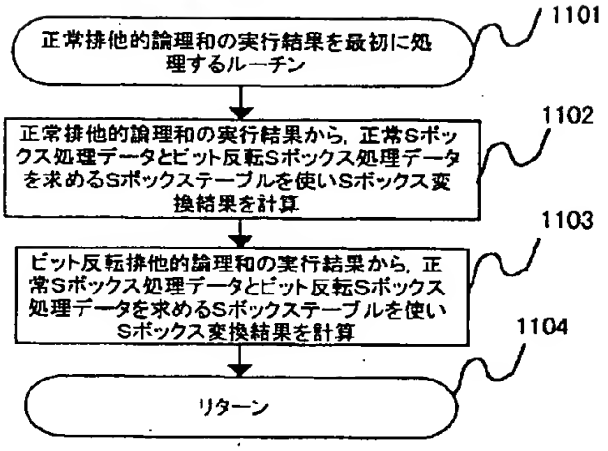
図29

2901				2903
	1	2	3	4
1	14	4	13	1
2	0	15	7	4
3	4	1	14	8
4	15	12	8	2

2902

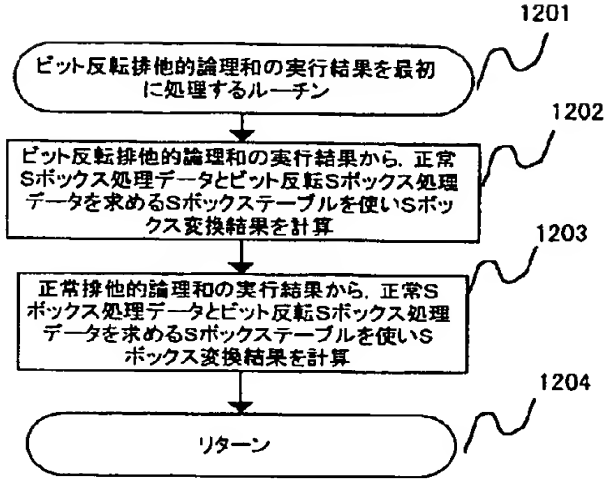
【図１１】

図11



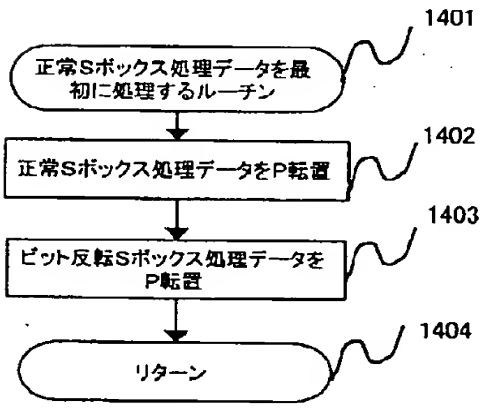
【図１２】

図12



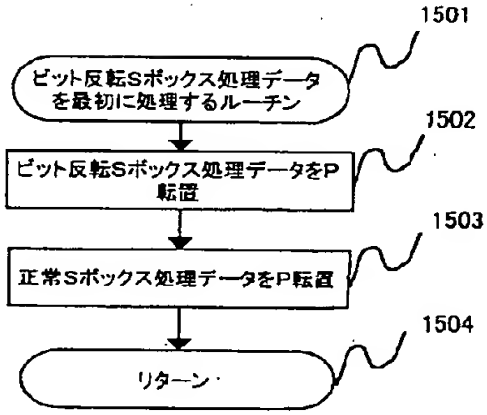
【図１４】

図14



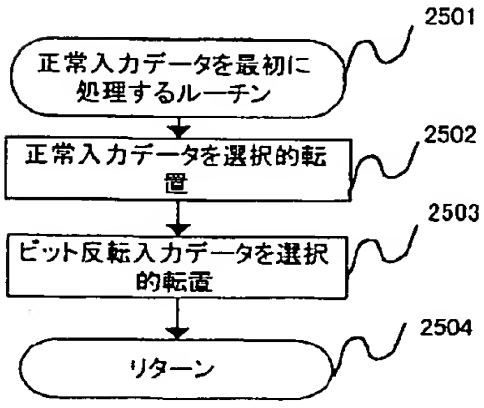
【図１５】

図15



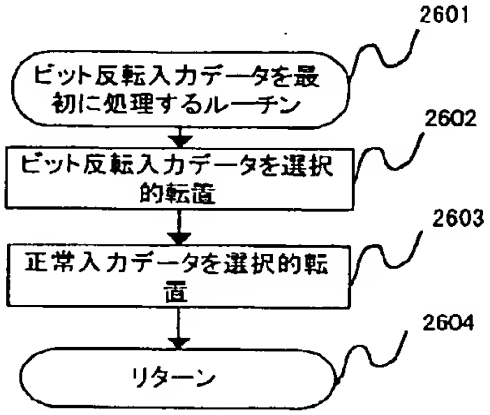
【図２５】

図25



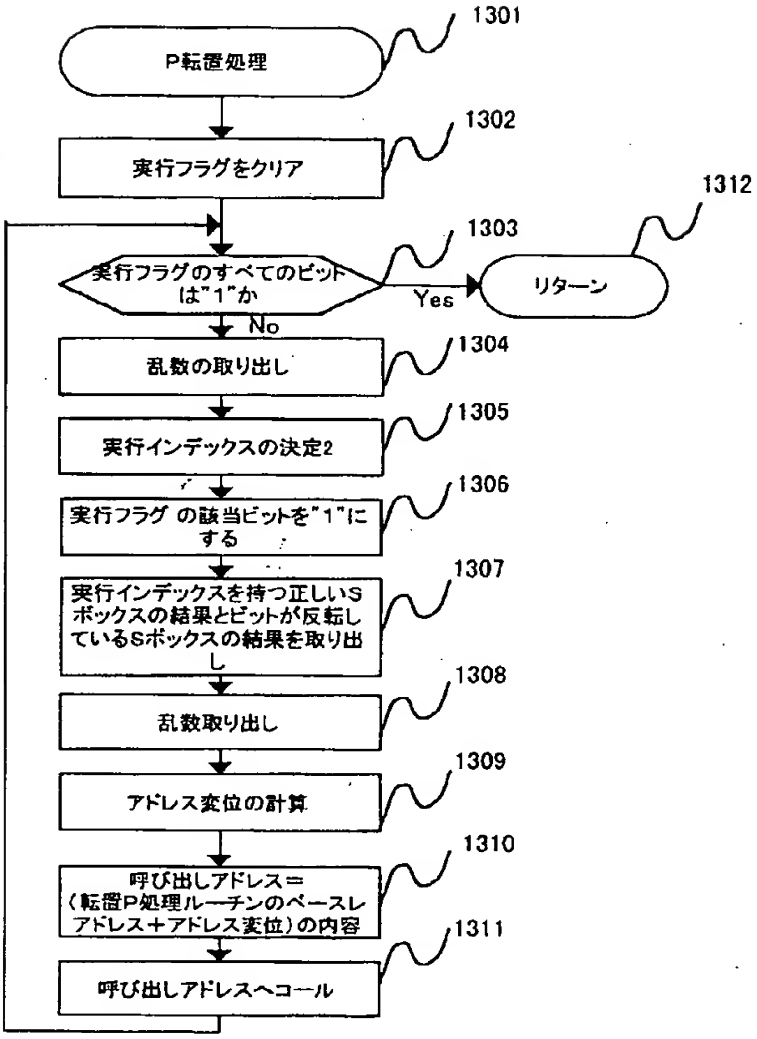
【図２６】

図26



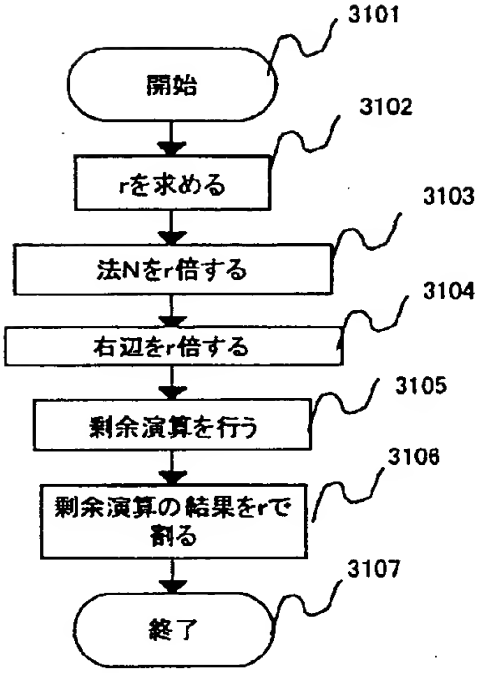
【図１３】

図13



【図３１】

図31



【図３０】

図30

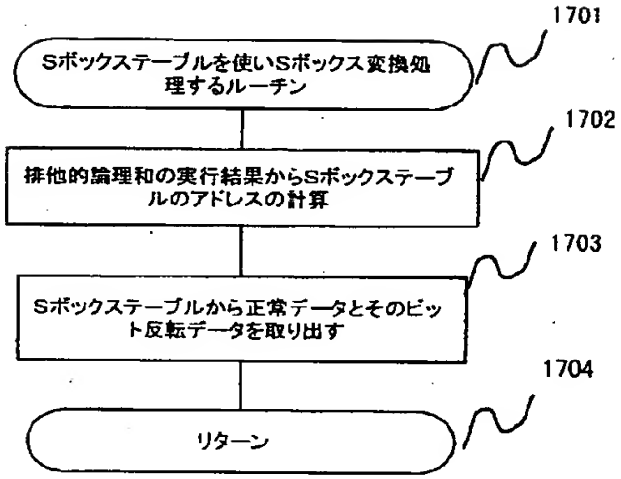
3001　3004　3005　3003

			2		3		4	
1	14	1	4	11	13	2	1	14
2	0	15	15	0	7	8	4	10
3	4	11	1	14	14	1	8	7
4	15	0	12	3	8	7	2	13

3002

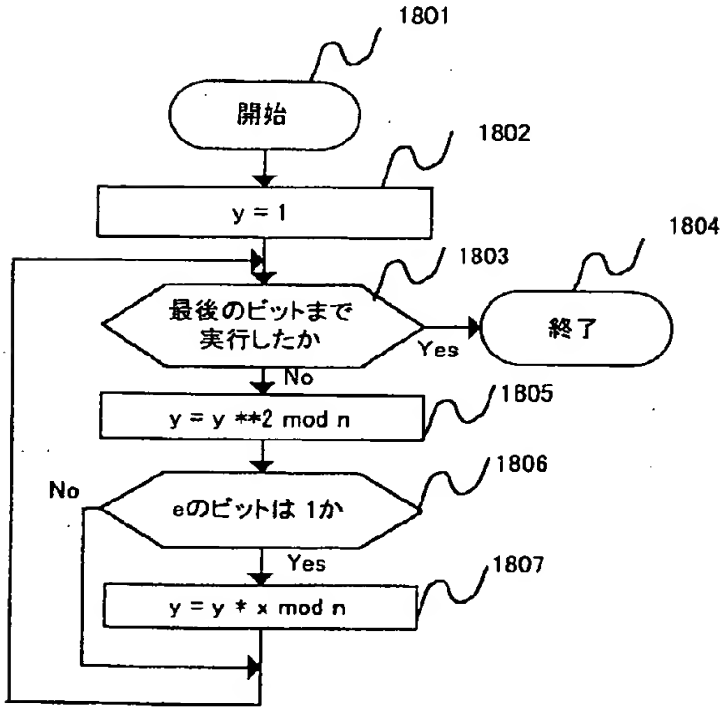
【図１７】

図17



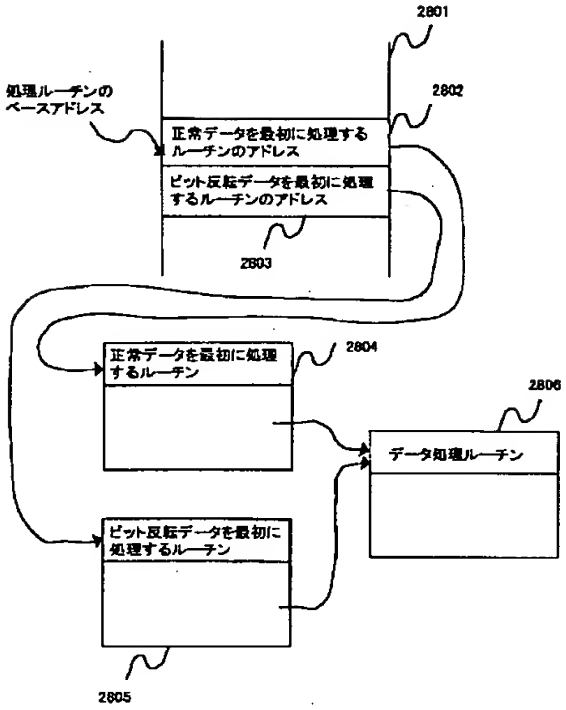
【図１８】

図18



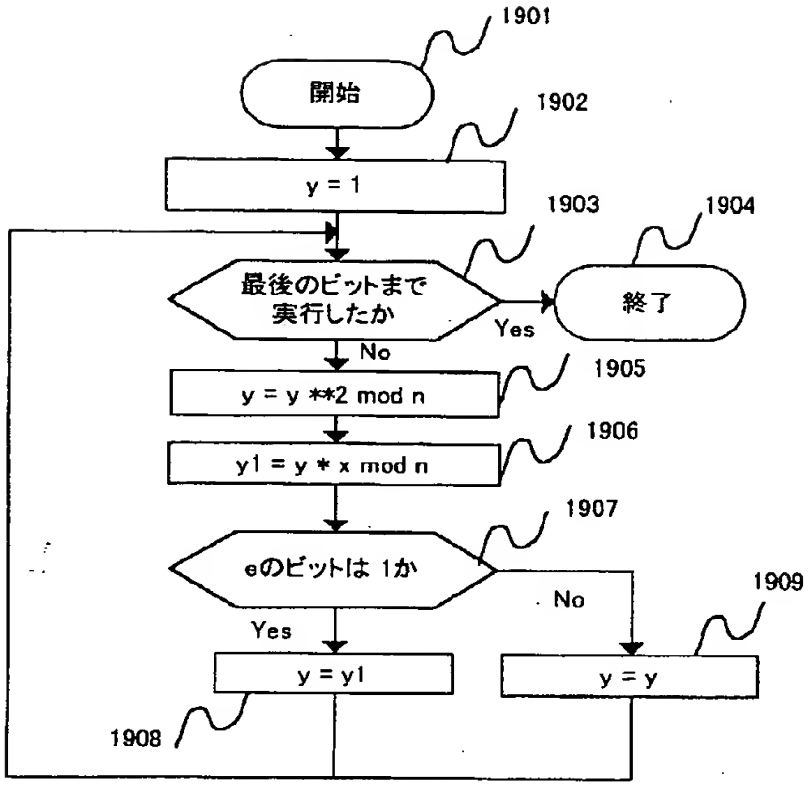
【図２８】

図28



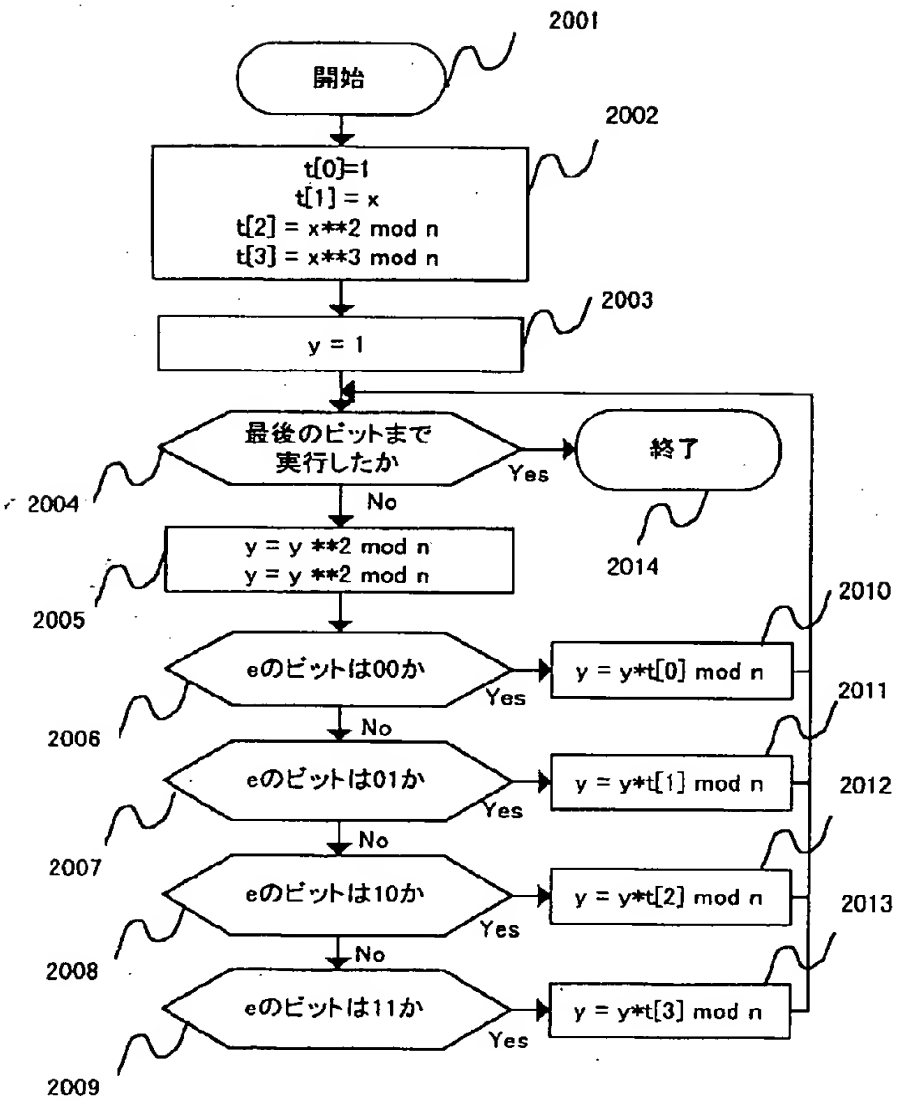
【図１９】

図19


1901
開始
1902
y = 1
1903
最後のビットまで
実行したか
Yes
1904
終了
No
1905
y = y **2 mod n
1906
y1 = y * x mod n
1907
eのビットは 1か
No
1909
Yes
y = y1
y = y
1908

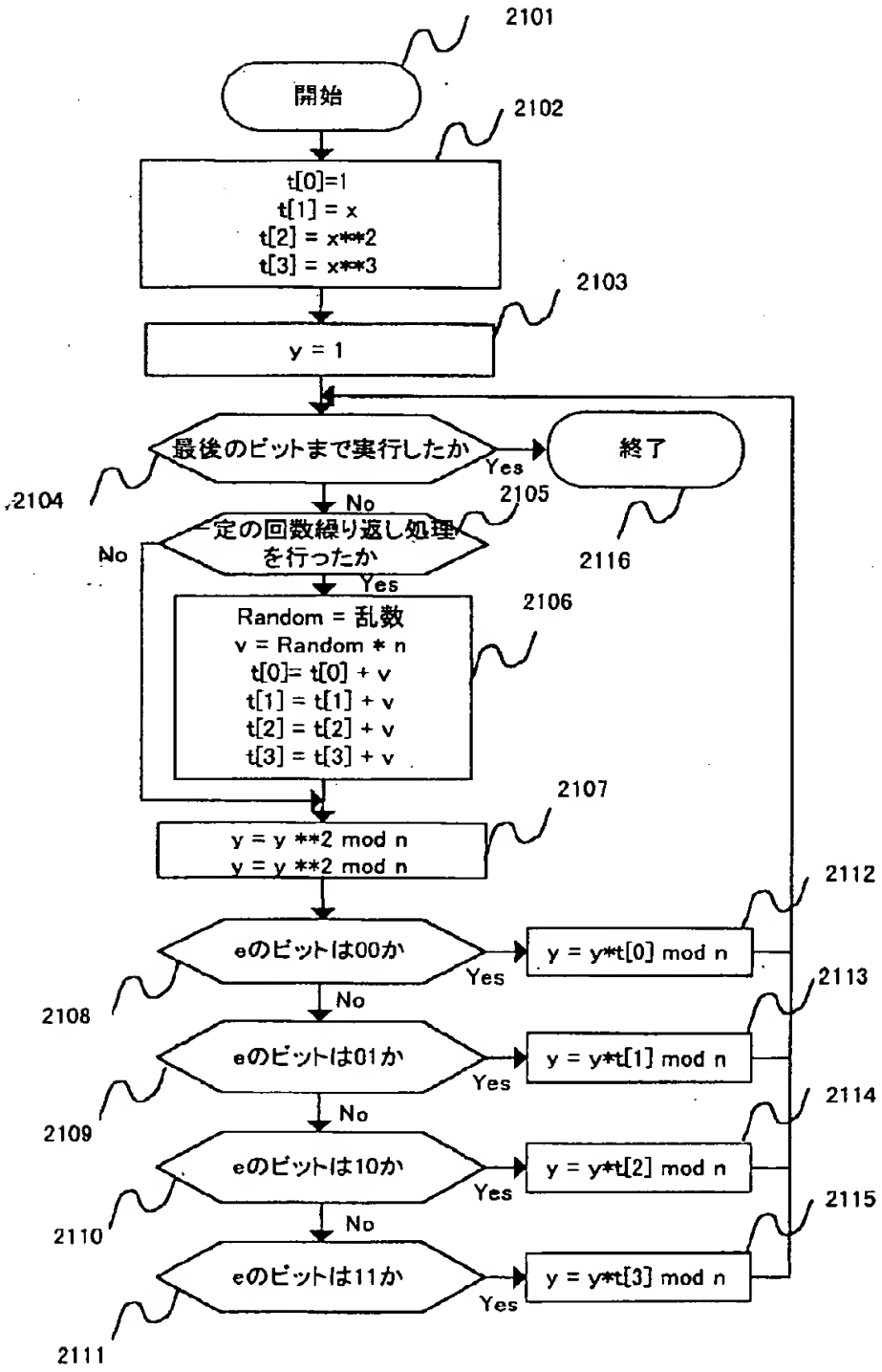
【図２０】

図20


2001
開始
2002
t[0]=1
t[1] = x
t[2] = x**2 mod n
t[3] = x**3 mod n
2003
y = 1
最後のビットまで
実行したか
Yes
終了
2004
No
2014
y = y **2 mod n
y = y **2 mod n
2005
2010
eのビットは00か
y = y*t[0] mod n
Yes
2011
No
2006
eのビットは01か
y = y*t[1] mod n
Yes
2012
No
2007
eのビットは10か
y = y*t[2] mod n
Yes
2013
No
2008
eのビットは11か
y = y*t[3] mod n
Yes
2009

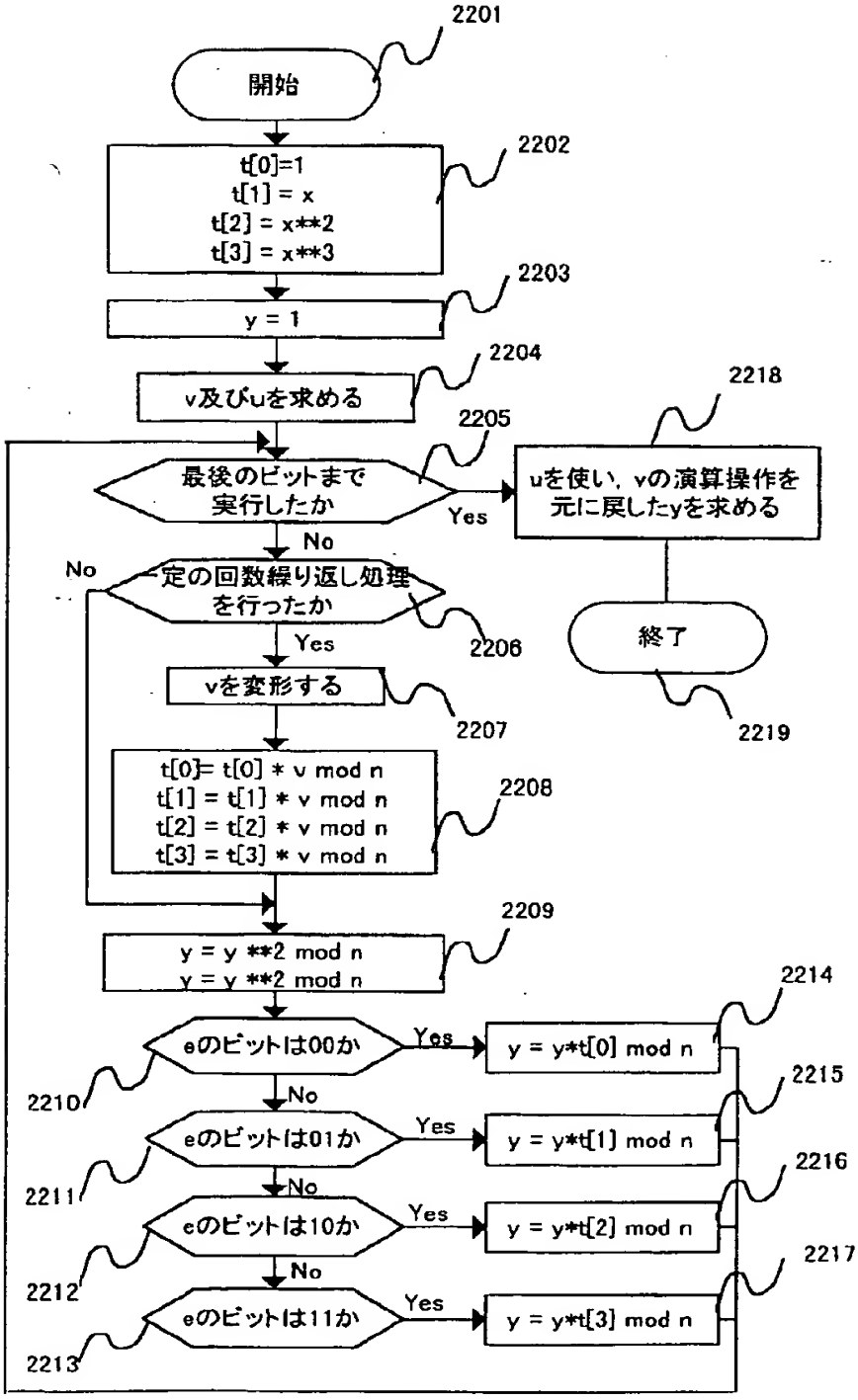
【図２１】

図21


2101
開始
2102
t[0]=1
t[1] = x
t[2] = x**2
t[3] = x**3
2103
y = 1
最後のビットまで実行したか
Yes
終了
2104
No
2105
一定の回数繰り返し処理を行ったか
No
Yes
2116
2106
Random = 乱数
v = Random * n
t[0]= t[0] + v
t[1] = t[1] + v
t[2] = t[2] + v
t[3] = t[3] + v
2107
y = y **2 mod n
y = y **2 mod n
2112
eのビットは00か
Yes
y = y*t[0] mod n
2108
No
2113
eのビットは01か
Yes
y = y*t[1] mod n
2109
No
2114
eのビットは10か
Yes
y = y*t[2] mod n
2110
No
2115
eのビットは11か
Yes
y = y*t[3] mod n
2111

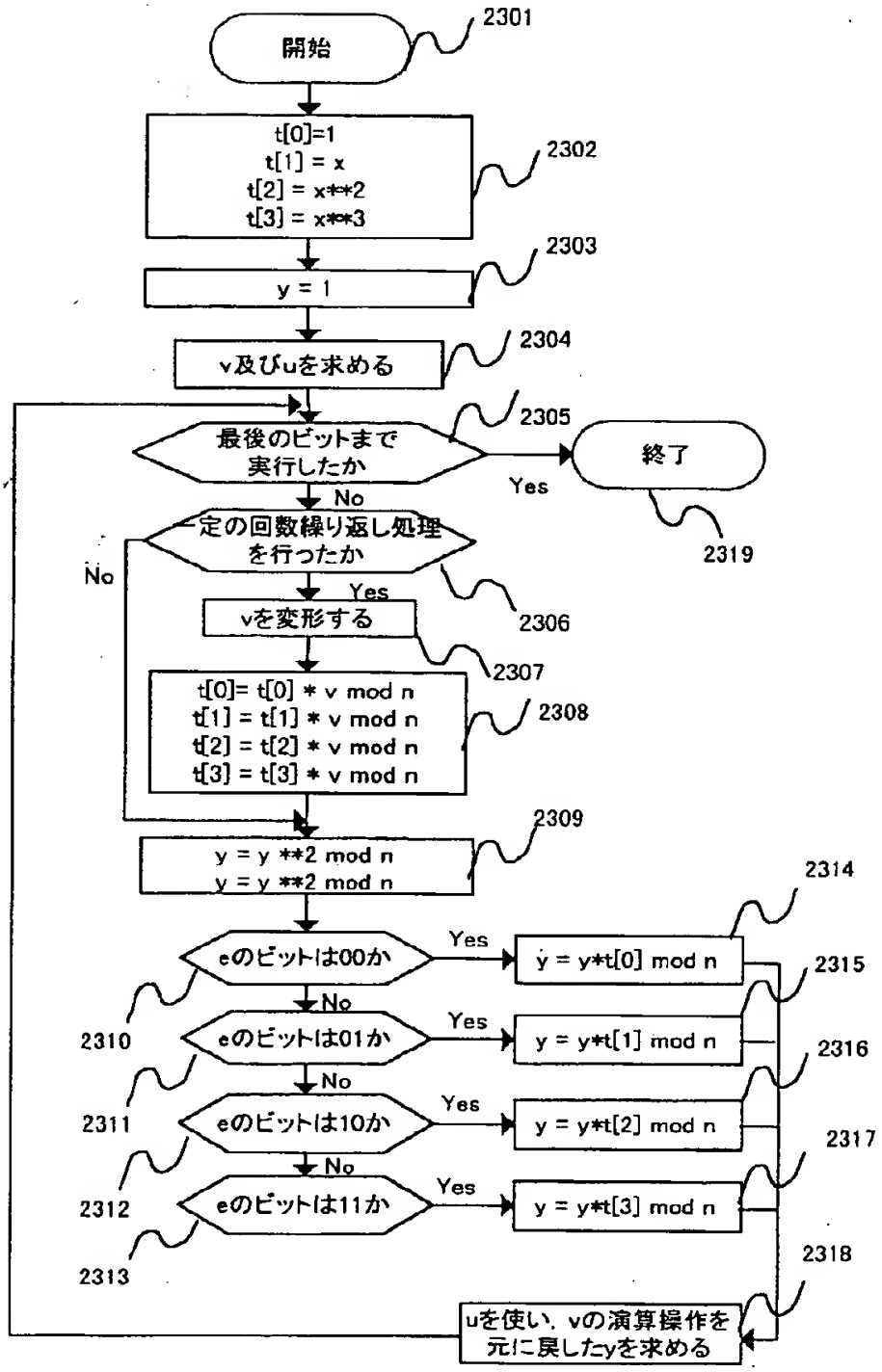
【図２２】

図22


2201
開始
2202
t[0]=1
t[1] = x
t[2] = x**2
t[3] = x**3
2203
y = 1
2204
v及びuを求める
2205
最後のビットまで実行したか
Yes
2218
uを使い，vの演算操作を元に戻したyを求める
終了
2219
No
No
一定の回数繰り返し処理を行ったか
2206
Yes
vを変形する
2207
t[0]= t[0] * v mod n
t[1] = t[1] * v mod n
t[2] = t[2] * v mod n
t[3] = t[3] * v mod n
2208
2209
y = y **2 mod n
y = y **2 mod n
eのビットは00か
Yes
y = y*t[0] mod n
2214
2210
No
eのビットは01か
Yes
y = y*t[1] mod n
2215
2211
No
eのビットは10か
Yes
y = y*t[2] mod n
2216
2212
No
eのビットは11か
Yes
y = y*t[3] mod n
2217
2213

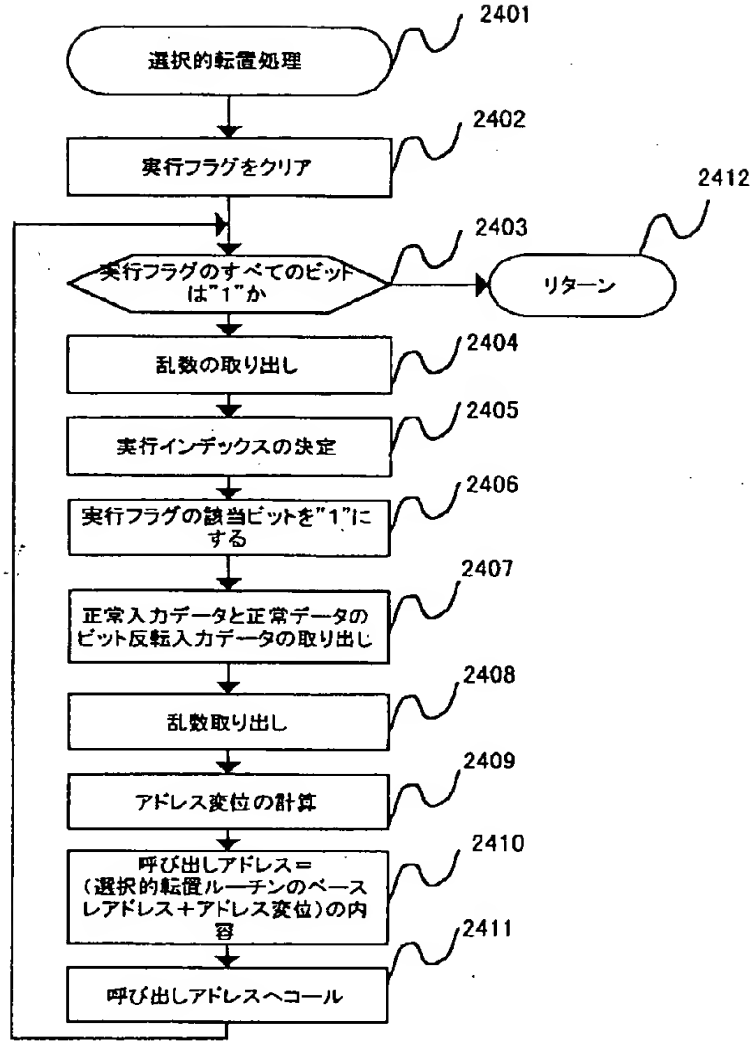
【図２３】

図23


開始
2301
t[0]=1
t[1] = x
t[2] = x**2
t[3] = x**3
2302
y = 1
2303
v及びuを求める
2304
最後のビットまで実行したか
2305
Yes
終了
2319
No
一定の回数繰り返し処理を行ったか
No
Yes
vを変形する
2306
2307
t[0]= t[0] * v mod n
t[1] = t[1] * v mod n
t[2] = t[2] * v mod n
t[3] = t[3] * v mod n
2308
y = y **2 mod n
y = y **2 mod n
2309
eのビットは00か
Yes
y = y*t[0] mod n
2314
2315
No
2310
eのビットは01か
Yes
y = y*t[1] mod n
2316
No
2311
eのビットは10か
Yes
y = y*t[2] mod n
2317
No
2312
eのビットは11か
Yes
y = y*t[3] mod n
2313
2318
uを使い、vの演算操作を元に戻したyを求める

【図２４】

図24



フロントページの続き

(72)発明者　奥原　進
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア事業部内

(72)発明者　神永　正博
東京都国分寺市東恋ケ窪一丁目280番地　株式会社日立製作所中央研究所内

Ｆターム(参考)　5B035 AA13 BB09 CA29 CA38
5J104 AA41 AA47 EA04 EA08 JA13 JA28 NA08 NA09 NA35 NA40 NA42